FreeBSD Porter’s Handbook

Содержание

1. Введение . . . 8
2. Making a New Port . . . 9
3. Быстрое портирование . . . 10
   3.1. Создание файла Makefile . . . 10
   3.2. Создание информационных файлов . . . 11
   3.3. Создание файла с контрольной суммой . . . 13
   3.4. Тестирование порта . . . 13
   3.5. Проверка вашего порта утилитой `portlint` . . . 14
   3.6. Посылка нового порта . . . 14
4. Медленное портирование . . . 16
   4.1. Как всё это работает . . . 16
   4.2. Получение исходного кода . . . 17
   4.3. Модификация порта . . . 18
   4.4. Создание патчей . . . 18
   4.5. Конфигурирование . . . 21
   4.6. Обработка пользовательского ввода . . . 21
5. Настройка файла Makefile . . . 22
   5.1. Оригинальные исходный код . . . 22
   5.2. Именование . . . 22
   5.3. Разделение по категориям . . . 28
   5.4. Дистрибутивные файлы . . . 37
   5.5. `MAINTAINER` . . . 48
   5.6. `COMMENT` . . . 48
   5.7. `PORTSCOUT` . . . 49
   5.8. Зависимости . . . 49
   5.9. `MASTERDIR` . . . 57
   5.10. Страницы Справочника . . . 58
   5.11. Файлы в формате info . . . 58
   5.12. Опции для Makefile . . . 58
   5.13. Задание рабочего каталога . . . 69
   5.14. Разрешение конфликтов . . . 70
   5.15. Установка файлов . . . 71
6. Особые соглашения . . . 75
   6.1. Staging . . . 75
   6.2. Динамические библиотеки . . . 76
   6.3. Порты с ограничениями на распространение или с правовым обременением . . . 76
   6.4. Механизмы построения . . . 79
   6.5. Использование GNU Autotools . . . 82

6.6. Использование GNU `gettext` . . . 84
6.7. Использование Perl . . . 86
6.8. Использование X11 . . . 87
6.9. Использование GNOME . . . 90
6.10. Использование Qt . . . 90
6.11. Использование KDE . . . 94
6.12. Использование Java . . . 96
6.13. Веб-приложения, Apache и PHP . . . 100
6.14. Использование Python . . . 103
6.15. Использование Tcl/Tk . . . 105
6.16. Использование Emacs . . . 106
6.17. Использование Ruby . . . 106
6.18. Использование SDL . . . 107
6.19. Использование wxWidgets . . . 108
6.20. Использование Lua . . . 113
6.21. Использование `iconv` . . . 118
6.22. Использование Xfce . . . 120
6.23. Использование Mozilla . . . 121
6.24. Использование баз данных . . . 122
6.25. Запуск и остановка служб (сценарии `rc`) . . . 123
6.26. Добавление пользователей и групп . . . 126
6.27. Порты, требующие наличия исходных текстов ядра . . . 126
7. Продвинутые практики pkg-plist . . . 127
7.1. Изменение содержимого pkg-plist в зависимости от make-переменных . . . 127
7.2. Пустые каталоги . . . 128
7.3. Конфигурационные файлы . . . 129
7.4. Динамический или статический список упаковки . . . 130
7.5. Автоматическое создание списка упаковки . . . 130
8. Файлы pkg-* . . . 131
8.1. pkg-message . . . 131
8.2. pkg-install . . . 131
8.3. pkg-deinstall . . . 131
8.4. Изменение имён файлов pkg-* . . . 132
8.5. Использование `SUB_FILES` и `SUB_LIST` . . . 132
9. Тестирование вашего порта . . . 134
9.1. Запуск `make describe` . . . 134
9.2. Portlint . . . 134
9.3. Port Tools . . . 134
9.4. `PREFIX` и `DESTDIR` . . . 135
9.5. Tinderbox . . . 136
9.6. Poudriere . . . 136

10. Обновление отдельного порта . . . 137
10.1. Использование Subversion для создания патчей . . . 139
10.2. Файлы UPDATING и MOVED . . . 140
11. Безопасность портов . . . 142
11.1. Почему безопасность так важна . . . 142
11.2. Исправление уязвимостей безопасности . . . 142
11.3. Обеспечение сообщества информацией . . . 143
12. Что делать нужно, и что делать нельзя . . . 149
12.1. Введение . . . 149
12.2. `WRKDIR` . . . 149
12.3. `WRKDIRPREFIX` . . . 149
12.4. Различение операционных систем и версий ОС . . . 149
12.5. Написание чего-либо после bsd.port.mk . . . 150
12.6. Использование выражения `exec` в сценариях обёртках . . . 151
12.7. Поступайте разумно . . . 151
12.8. Работа как с `CC`, так и `CXX` . . . 151
12.9. Использование `CFLAGS` . . . 152
12.10. Библиотеки потоков . . . 153
12.11. Пожелания . . . 153
12.12. README.html . . . 153
12.13. Пометка неустанавливаемого порта как `BROKEN`, `FORBIDDEN` или `IGNORE` . . . 153
12.14. Пометка порта на удаление с `DEPRECATED` или `EXPIRATION_DATE` . . . 156
12.15. Избегайте использования конструкции `.error` . . . 156
12.16. Использование sysctl . . . 157
12.17. Меняющиеся дистрибутивные файлы . . . 157
12.18. Избегание линуксизмов . . . 157
12.19. Разное . . . 158
13. Примерный Makefile . . . 159
14. Актуализация . . . 161
14.1. FreshPorts . . . 161
14.2. Web-интерфейс к хранилищу исходных текстов . . . 161
14.3. Список рассылки FreeBSD, посвящённый портам . . . 161
14.4. Кластер построения портов FreeBSD . . . 162
14.5. Portscout: сканер дистрибутивных файлов портов FreeBSD . . . 162
14.6. Система мониторинга портов FreeBSD . . . 162
15. Значения `USES` . . . 164
15.1. An Introduction to `USES` . . . 164
15.2. `7z` . . . 164
15.3. `ada` . . . 165
15.4. `autoreconf` . . . 165
15.5. `blaslapack` . . . 165

15.6. bdb . . . 165
15.7. bison . . . 166
15.8. cabal . . . 166
15.9. cargo . . . 167
15.10. charsetfix . . . 167
15.11. cmake . . . 167
15.12. compiler . . . 168
15.13. cpe . . . 168
15.14. cran . . . 169
15.15. desktop-file-utils . . . 169
15.16. desthack . . . 169
15.17. display . . . 169
15.18. dos2unix . . . 169
15.19. drupal . . . 170
15.20. eigen . . . 170
15.21. fakeroot . . . 170
15.22. fam . . . 170
15.23. firebird . . . 170
15.24. fonts . . . 170
15.25. fortran . . . 171
15.26. fuse . . . 171
15.27. gem . . . 171
15.28. gettext . . . 171
15.29. gettext-runtime . . . 171
15.30. gettext-tools . . . 171
15.31. ghostscript . . . 171
15.32. gl . . . 172
15.33. gmake . . . 172
15.34. gnome . . . 172
15.35. go . . . 175
15.36. gperf . . . 176
15.37. grantlee . . . 176
15.38. groff . . . 176
15.39. gssapi . . . 176
15.40. horde . . . 177
15.41. iconv . . . 178
15.42. imake . . . 178
15.43. kde . . . 178
15.44. kmod . . . 178
15.45. lha . . . 179
15.46. libarchive . . . 179

15.47. libedit . . . 179
15.48. libtool . . . 179
15.49. linux . . . 179
15.50. localbase . . . 181
15.51. lua . . . 182
15.52. lxqt . . . 182
15.53. makeinfo . . . 182
15.54. makeself . . . 182
15.55. mate . . . 182
15.56. meson . . . 183
15.57. metaport . . . 183
15.58. mysql . . . 183
15.59. mono . . . 184
15.60. motif . . . 184
15.61. ncurses . . . 184
15.62. ninja . . . 184
15.63. objc . . . 184
15.64. openal . . . 185
15.65. pathfix . . . 185
15.66. pear . . . 185
15.67. perl5 . . . 185
15.68. pgsql . . . 185
15.69. php . . . 186
15.70. pkgconfig . . . 188
15.71. pure . . . 188
15.72. pyqt . . . 188
15.73. python . . . 189
15.74. qmail . . . 190
15.75. qmake . . . 190
15.76. qt . . . 190
15.77. qt-dist . . . 190
15.78. readline . . . 191
15.79. samba . . . 191
15.80. scons . . . 191
15.81. shared-mime-info . . . 192
15.82. shebangfix . . . 192
15.83. sqlite . . . 194
15.84. ssl . . . 194
15.85. tar . . . 195
15.86. tcl . . . 195
15.87. terminfo . . . 195

15.88. tk . . . 196
15.89. uidfix . . . 196
15.90. uniquefiles . . . 196
15.91. varnish . . . 196
15.92. webplugin . . . 196
15.93. xfce . . . 197
15.94. xorg . . . 197
15.95. xorg-cat . . . 199
15.96. zip . . . 199
16. Значения __FreeBSD_version . . . 200
16.1. FreeBSD 13 Versions . . . 200
16.2. FreeBSD 12 Versions . . . 213
16.3. FreeBSD 11 Versions . . . 228
16.4. FreeBSD 10 Versions . . . 247
16.5. FreeBSD 9 Versions . . . 261
16.6. FreeBSD 8 Versions . . . 269
16.7. FreeBSD 7 Versions . . . 286
16.8. FreeBSD 6 Versions . . . 295
16.9. FreeBSD 5 Versions . . . 301
16.10. FreeBSD 4 Versions . . . 313
16.11. FreeBSD 3 Versions . . . 318
16.12. FreeBSD 2.2 Versions . . . 319
16.13. FreeBSD 2 Before 2.2-RELEASE Versions . . . 320

Глава 1. Введение

Коллекция портов FreeBSD является способом, используемым практически каждым для установки приложений ("портов") на FreeBSD. Как и почти всё остальное во FreeBSD, эта система в основном является добровольно поддерживаемым начинанием. Важно иметь это в виду при чтении данного документа.

Во FreeBSD каждый может прислать новый порт либо изъявить желание поддерживать существующий порт, если его никто ещё никто не поддерживает-вам не нужно иметь никаких особых привилегий на внесение изменений, чтобы это делать.

Глава 2. Making a New Port

Итак, вы интересуетесь, как создать собственный порт или обновить существующий? Великолепно!

Ниже находятся некоторые указания по созданию нового порта для FreeBSD. Если вы хотите обновить существующий порт, вы должны прочесть их, а затем Обновление отдельного порта.

Если этот документ недостаточно подробен, вы должны обратиться к файлу /usr/ports/Mk/bsd.port.mk, который включается в make-файл каждого порта. Он хорошо прокомментирован, и даже если вы не занимаетесь хакингом make-файлов каждодневно, из него вы сможете узнать много нового. Кроме того, конкретные вопросы можно задать, послав письмо на адрес Список рассылки, посвящённый Портам FreeBSD.

Только часть переменных (`VAR`), которые могут быть переопределены, описаны в этом документе. Большинство (если не все) описаны в начале файла /usr/ports/Mk/bsd.port.mk; остальные, скорее всего, тоже там описаны. Заметьте, что в этом файле используется нестандартная настройка шага табуляции: Emacs и Vim должны распознать это при загрузке файла. Как vi(1), так и ex(1) могут быть настроены на использование правильного значения выдачей команды `:set tabstop=4` после загрузки файла.

Ищете, с чего бы начать попроще? Посмотрите на перечень запрошенных портов, есть ли там такие, над которыми вы можете работать.

Глава 3. Быстрое портирование

В этом разделе описано, как создать новый порт на скорую руку. Во многих случаях этого бывает не достаточно, так что вам нужно будет прочитать документ дальше.

Во-первых, скачайте оригинальный tar-файл и поместите его в каталог `DISTDIR`, который по умолчанию есть не что иное, как /usr/ports/distfiles.

> Здесь предполагается, что программное обеспечение компилируется без проблем как есть, то есть для работы приложения на вашей системе FreeBSD не потребовалось абсолютно никаких изменений. Если требовалось что-то изменить, то вам придется обратиться также и к следующему разделу.

> Перед началом портирования рекомендуется установить переменную make(1) `DEVELOPER` в /etc/make.conf.
>
> ```
> # echo DEVELOPER=yes >> /etc/make.conf
> ```
>
> Эта настройка включает "режим разработчика", в котором отображаются предупреждения при использовании устаревших конструкций и задействуются некоторые дополнительные проверки при вызове команды `make`.

3.1. Создание файла Makefile

Минимальный Makefile будет выглядеть примерно так:

```
# $FreeBSD$

PORTNAME=   oneko
PORTVERSION=    1.1b
CATEGORIES= games
MASTER_SITES=   ftp://ftp.cs.columbia.edu/archives/X11R5/contrib/

MAINTAINER= youremail@example.com
COMMENT=    Cat chasing a mouse all over the screen

.include <bsd.port.mk>
```

> В некоторых случаях в заголовке Makefile существующего порта могут содержаться дополнительные строки, такие как название порта и дата его создания. Эта дополнительная информация была объявлена устаревшей и находится в процессе удаления.

Посмотрим, сможете ли вы его понять. Не обращайте внимание на содержимое строчки

`$FreeBSD$`, она будет заполнена автоматически системой Subversion, когда порт будет импортирован в наше дерево портов. Вы можете найти более подробный пример в разделе пример Makefile.

3.2. Создание информационных файлов

Имеется два информационных файла, которые требуются для любого порта, вне зависимости от того, является ли он пакетом или нет. Это pkg-descr и pkg-plist. Префикс pkg- отличает их от других файлов.

3.2.1. pkg-descr

Это более подробное краткое описание порта. От одного до нескольких абзацев, кратко описывающих, что представляет собой порт, будет достаточно.

> Это *не* руководство и не подробнейшее описание того, как использовать или компилировать порт! *Пожалуйста, будьте внимательны при копировании текста из README или страниц справочника*; слишком часто они не являются кратким описанием порта или имеют неудобный формат (например, страницы справочника выровнены пробелами, что особенно плохо смотрится с моноширинными шрифтами).

Хорошо составленный pkg-descr описывает порт достаточно полно, чтобы пользователю не приходилось сверяться с документацией или посещать вебсайт для понимания того, что делает данное программное обеспечение, чем оно может быть полезно или какие хорошие функции у него имеются. Упоминание про определённые требования, такие как используемый графический инструментарий, тяжёлые зависимости, окружение для запуска или используемый язык программирования помогут пользователям определиться, будет ли этот порт для них работать.

Включите сюда URL официальной домашней страницы Интернет. Перед *одним* из сайтов (выберите основной) добавьте `WWW:` (с последующим единичным пробелом) для того, чтобы вспомогательные утилиты работали правильно. Если URI является корнем сайта или каталогом, то значение должно быть дополнено косой чертой.

> Если указанная для порта веб-страница не доступна, попытайтесь сперва поискать, был ли официальный сайт перемещён, переименован или размещён в другом месте.

Следующий пример показывает, как должен выглядеть ваш pkg-descr:

```
This is a port of oneko, in which a cat chases a poor mouse all over
the screen.
 :
(etc.)

WWW: http://www.oneko.org/
```

3.2.2. pkg-plist

Здесь перечисляются все файлы, устанавливаемые портом. Его также называют "списком для упаковки", потому что пакет генерируется упаковкой файлов, которые здесь указаны. Имена путей указываются относительно установочного префикса (обычно /usr/local). Если порт во время установки создает каталоги, убедитесь, что добавлены строки `@dirrm` для удаления каталогов при удалении пакета.

Вот маленький пример:

```
bin/oneko
man/man1/oneko.1.gz
lib/X11/app-defaults/Oneko
lib/X11/oneko/cat1.xpm
lib/X11/oneko/cat2.xpm
lib/X11/oneko/mouse.xpm
@dirrm lib/X11/oneko
```

Обратитесь к странице справочной системы по команде pkg-create(8) с подробным описанием формата списка упаковки.

> Рекомендуется, чтобы имена файлов в этом списке были отсортированы в алфавитном порядке. Это позволит значительно облегчить сверку изменений при обновлении порта.

> Создание списка упаковки вручную может оказаться весьма трудоёмкой задачей. Если порт устанавливает большое количество файлов, раздел об автоматическом построении списка упаковки может помочь сэкономить время.

Существует только одно исключение, когда у порта может отсутствовать pkg-plist. Если порт устанавливает лишь несколько файлов, а возможно, и каталогов, то они могут быть перечислены в переменных `PLIST_FILES` и `PLIST_DIRS`, соответственно, внутри файла Makefile порта. К примеру, мы можем обойтись без файла pkg-plist у приведённого выше порта oneko, добавив следующие строки в Makefile:

```
PLIST_FILES=    bin/oneko \
        man/man1/oneko.1.gz \
        lib/X11/app-defaults/Oneko \
        lib/X11/oneko/cat1.xpm \
        lib/X11/oneko/cat2.xpm \
        lib/X11/oneko/mouse.xpm
PLIST_DIRS= lib/X11/oneko
```

Конечно, переменная `PLIST_DIRS` не должна задаваться, если порт не устанавливает никаких каталогов.

Несколько портов могут совместно использовать общий каталог. В этом случае PLIST_DIRS следует заменить на PLIST_DIRSTRY, так чтобы каталог удалялся только если он пуст, а иначе игнорировался. Использование PLIST_DIRS и PLIST_DIRSTRY аналогично @dirrm и @dirrmtry в pkg-plist, описание которых входит в Очистка пустых каталогов.

Обратной стороной такого способа перечисления файлов и каталогов порта является невозможность использования последовательностей команд, описанных в pkg-create(8). Поэтому он подходит для простых портов, что делает их ещё более простыми. Одновременно с этим положительным моментом является уменьшение количества файлов в коллекции портов. Пожалуйста, подумайте над использованием этой техники, прежде чем создавать pkg-plist.

Далее мы увидим, как можно использовать файлы pkg-plist и PLIST_FILES выполнения более сложных задач.

3.3. Создание файла с контрольной суммой

Просто введите команду make makesum. Правила утилиты make автоматически сгенерируют файл distinfo.

Если у извлекаемого файла регулярно меняется контрольная сумма и вы не сомневаетесь в надежности источника (т.е. он получен из CD производителя, либо ежедневно обновляется документация), то вы должны указать эти файлы в переменной IGNOREFILES. Тогда контрольная сумма при выполнении make makesum для этого файла создаваться не будет, а вместо этого для него будет установлено значение IGNORE.

3.4. Тестирование порта

Вы должны удостовериться, что правила построения порта выполняют именно то, что вы хотите, включая создание пакета для порта. Вот те важные вещи, которые вы должны проверить.

- pkg-plist не содержит ничего сверх того, что устанавливается портом
- pkg-plist содержит абсолютно все, что устанавливается портом
- Порт может быть установлен с помощью указания цели install. Это позволяет убедиться в правильной работе сценария установки.
- Порт может быть правильным образом удалён с помощью указания цели deinstall. Это позволяет убедиться в правильной работе сценария удаления.
- Следует убедиться, что make package можно запустить из-под обычного пользователя (то есть, не из-под root). Если это не так, в Makefile порта должно быть добавлено NEED_ROOT=yes.

Procedure: Рекомендуемый порядок проверки

1. make stage

2. `make check-orphans`
3. `make package`
4. `make install`
5. `make deinstall`
6. `pkg add package-filename`
7. `make package` (из-под пользователя)

Убедитесь, что на любом из этапов не выдается никаких предупреждений.

Основательное автоматизированное тестирование может быть выполнено при помощи ports-mgmt/tinderbox или ports-mgmt/poudriere из Коллекции Портов. Эти приложения используют `jails`, в которых проверяются все перечисленные выше этапы без изменения состояния основной системы.

3.5. Проверка вашего порта утилитой `portlint`

Будьте добры, пользуйтесь утилитой `portlint` для проверки того, что ваш порт соответствует нашим рекомендациям. Программа ports-mgmt/portlint является частью Коллекции Портов. В частности, вы можете захотеть проверить, правильно ли сформирован файл Makefile и соответствующим ли образом именован пакет.

3.6. Посылка нового порта

Перед посылкой нового порта прочитайте раздел о том, что можно и нельзя делать.

Когда вы наконец довольны своим первым портом, единственное, что осталось сделать, это включить его в основное дерево портов FreeBSD и осчастливить этим всех остальных. Нам не нужен ни каталог work, ни пакет pkgname.tgz, так что удалите их прямо сейчас.

Затем получите файл shar(1). Предполагая, что порт называется oneko, перейдите в каталог выше, где находится каталог `oneko`, и наберите: `shar find oneko > oneko.shar`

Включите oneko.shar в сообщение об ошибке и пошлите его с помощью send-pr(1). Обратитесь к разделу Сообщения об ошибках и общие замечания для получения подробной информации о send-pr(1)).

Укажите в сообщении категорию `ports` и класс `change-request`. *Не* указывайте, что сообщение имеет статус `confidential`! Добавьте краткое описание программы в поле "Description" отправляемого PR (например, содержимое `COMMENT` в сокращённом варианте) и сам файл в виде архива .shar в поле "Fix".

Хорошее описание в заголовке сообщения о проблеме значительно облегчает работу коммиттеров портов. Для новых портов мы предпочитаем нечто вроде "New port: <категория>/<название порта> <краткое описание порта>". Следование этой схеме упрощает и ускоряет начало работы по

> добавлению нового порта.

Повторим ещё раз, что *не нужно включать ни оригинальный файл с дистрибутивом, ни каталог work, ни пакет, построенный вами командой* `make package`; для новых портов используйте shar(1), но не diff(1).

После отправки порта, пожалуйста, потерпите. Время, необходимое для включения нового порта во FreeBSD, может занимать от нескольких дней до нескольких месяцев. Здесь можно увидеть список ожидающих PR для портов.

После рассмотрения нового порта мы при необходимости вам ответим, а затем включим порт в наше дерево. Ваше имя также будет добавлено в список Дополнительных контрибуторов проекта FreeBSD и другие файлы.

Глава 4. Медленное портирование

Итак, все оказалось не так уж и просто, и порт потребовал некоторых модификаций для того, чтобы заставить его работать. В этом разделе мы расскажем, шаг за шагом, как его модифицировать, чтобы он работал с нашей системой портов.

4.1. Как всё это работает

Во-первых, когда пользователь дает в своем каталоге с портом команду `make`, происходит целая череда событий. Во время чтения этого текста может оказаться полезным иметь файл bsd.port.mk открытым в другом окне, что сильно поможет в их понимании.

Но не волнуйтесь сильно, если вы не до конца понимаете, что делается в bsd.port.mk, не так уж много людей его понимает... *:→*

1. Запускается цель `fetch`. Цель `fetch` отвечает за то, что архив исходных текстов имеется в наличии локально в каталоге `DISTDIR`. Если цель `fetch` не может найти требуемые файлы в каталоге `DISTDIR`, то они будут искаться по указателю URL `MASTER_SITES`, который устанавливается в Makefile, а также на наших FTP зеркалах, куда мы по возможности помещаем дистрибутивные файлы для архива. Затем она попытается сгрузить указанный файл с помощью `FETCH`, полагая, что запрашивающая машина имеет прямое подключение к Интернет. Если файл скачается удачно, то он будет помещен в каталог `DISTDIR` для последующего использования и обработки.
2. Выполняется цель `extract`. Она ищет дистрибутивный файл порта (как правило, tar-архив `gzip`) в каталоге `DISTDIR` и распаковывает его во временный каталог, задаваемый переменной `WRKDIR` (по умолчанию work).
3. Выполняется цель `patch`. Во-первых, применяются все патчи, заданные переменной `PATCHFILES`. Во-вторых, если какие-либо файлы с патчами, носящие имена patch-*, имеются в подкаталоге `PATCHDIR` (по умолчанию это каталог files), то они применяются в этот момент в алфавитном порядке.
4. Запускается цель `configure`. Здесь может выполняться любая из многих различных вещей.
   a. Если существует скрипт scripts/configure, то он запускается.
   b. Если задана переменная `HAS_CONFIGURE` или `GNU_CONFIGURE`, то запускается скрипт WRKSRC/configure.
5. Выполняется цель `build`. Она отвечает за переход в собственный рабочий каталог порта (`WRKSRC`) и его построение.
6. Выполняется цель `stage`. Конечный набор построенных файлов помещается во временный каталог (`STAGEDIR`, смотрите Staging). Иерархия этого каталога отражает иерархию каталогов системы, в которую данный пакет будет устанавливаться.
7. Выполняется цель `install`. В систему копируются файлы, перечисленные в pkg-plist порта.

Выше перечислены стандартные действия. Кроме того, вы сами можете определить цели `pre-что-то` или `post-что-то`, или создать скрипты с такими именами в подкаталоге scripts, и они будут запущены до или после выполнения действий по умолчанию.

Например, если у вас есть цель `post-extract`, определённая в вашем файле Makefile и файл pre-build в подкаталоге scripts, то после выполнения обычных действий по распаковке, будет вызвана цель `post-extract` а скрипт pre-build будет выполнен перед запуском стандартных правил построения. Рекомендуется использовать цели из Makefile, если действия достаточно просты, потому что в дальнейшем будет проще определить, какие нестандартные действия требует порт.

Действия по умолчанию выполняются целями `do-что-то` из bsd.port.mk. Например, команды для распаковки порта находятся в цели `do-extract`. Если вам не хватает цели по умолчанию, вы можете ее исправить, переопределив цель `do-something` в вашем файле Makefile.

> "Основные" цели (к примеру, `extract`, `configure` и так далее) не делают ничего больше, чем проверяют успешность завершения всех предыдущих шагов и вызывают настоящие цели или скрипты, и их не нужно менять. Если вам нужно изменить распаковку, исправляйте `do-extract`, но никогда не меняйте способ работы `extract`! Кроме того, цель `post-deinstall` является недействительной и не выполняется инфраструктурой портов.

Теперь, когда вы представляете, что происходит, когда пользователь набирает команду `make install`, давайте пройдемся через шаги, рекомендуемые для создания настоящего порта.

4.2. Получение исходного кода

Получите оригинальные исходные тексты (обычно) в виде упакованного tar-архива (foo.tar.gz или foo.tar.bz2) и скопируйте его в каталог `DISTDIR`. Всегда используйте исходные тексты *основной ветки разработки* везде, где это возможно.

Вам потребуется задать значение переменной `MASTER_SITES` так, чтобы оно указывало на местоположение оригинального tar-архива. В файле bsd.sites.mk вы найдёте краткие обозначения для большинства популярных сайтов. Пожалуйста, используйте эти сайты-и соответствующие определения-везде, где это возможно, чтобы избежать проблем повторения одной и той же информации в базе источников. Так как эти сайты со временем меняются, для всех причастных поддержка становится настоящим кошмаром.

Если вы не можете найти FTP/HTTP сайт с хорошим подключением к сети, или находите только сайты, которые имеют раздражающе нестандартные форматы, то можете захотеть поместить копию на надежный сервер FTP или HTTP, который вам доступен (например, ваша домашняя страница).

Если вы не можете найти доступного и надёжного места для помещения дистрибутивного файла, то мы сами сможем разместить его на сервере `ftp.FreeBSD.org`; однако это наименее рекомендуемое решение. Дистрибутивный файл должен быть помещён в каталог ~/public_distfiles/ одного из пользователей машины `freefall`. Попросите того, кто коммиттил ваш порт, сделать это. Этот человек также задаст переменной `MASTER_SITES` значение

MASTER_SITE_LOCAL, а в переменной MASTER_SITE_SUBDIR укажет своё имя пользователя с машины freefall.

Если дистрибутивные файлы вашего порта постоянно меняются по неизвестным причинам без изменения версий со стороны автора, остаётся только поместить дистрибутив на вашу домашнюю Web-страницу и указать её первой в списке MASTER_SITES. Если можете, попытайтесь договориться с автором порта об этом; это действительно помогает в достижении некоторого управления исходным кодом. Размещение собственной версии поможет избежать появления ошибок у пользователей типа checksum mismatch, а также уменьшит нагрузку на людей, сопровождающих наш FTP-сервер. Также, если у порта имеется только один основной сервер, то рекомендуется поместить архивную копию на свой сайт и указать его в списке MASTER_SITES вторым.

Если вашему порту требуются дополнительные патчи, доступные в Интернет, скачайте также и их, поместив в каталог DISTDIR. Не волнуйтесь, если они находятся не на том же сайте, откуда взят дистрибутивный архив, мы умеем обрабатывать такие ситуации (смотрите описание PATCHFILES ниже).

4.3. Модификация порта

Распакуйте копию дистрибутивного файла в отдельный каталог и внесите изменения, которые необходимы для того, чтобы порт компилировался нормально в текущей версии FreeBSD. *Тщательно отслеживайте* все, что вы делаете, этот процесс вам предстоит автоматизировать. Все, включая удаление, добавление или модификацию в файлах должны будут выполняться автоматически с помощью скриптов или файлов патчей, когда вы завершите работу над портом.

Если вашему порту во время компиляции, установки и настройки требуется довольно много взаимодействовать с пользователем, то посмотрите на один из классических скриптов Configure Лэрри Уолла (Larry Wall) и сделайте сами что-либо подобное. Предназначение новой коллекции портов - это сделать каждое приложение в стиле "plug-and-play" настолько, насколько это вообще возможно для конечного пользователя при минимальном использовании дискового пространства.

Если явно не указано обратное, то патчи, скрипты и другие файлы, которые вы создали и предоставили для Коллекции Портов FreeBSD, неявно подпадают под стандартные условия лицензии BSD.

4.4. Создание патчей

Файлы, которые добавлялись или изменялись в процессе создания порта, могут быть выявлены программой diff(1), а результат работы этой программы может быть в дальнейшем передан программе patch(1). Такое действие с обычным файлом подразумевает сохранение копии файла с первоначальным содержимым перед внесением каких-либо изменений.

```
% cp file file.orig
```

Патчи сохраняются в виде файлов с именем patch-*, где * обозначает путь к файлу, к которому применяется патч, такой как patch-Imakefile или patch-src-config.h.

После того как файл был изменён, используется diff(1) для получения разницы между первоначальной и изменённой версиями. Параметр -u указывает diff(1) выводить разницу в "унифицированном" формате, который также является предпочтительным.

```
% diff -u file.orig file > patch-pathname-file
```

Для порождении патчей для новых добавляемых файлов используется параметр -N, который заставляет diff(1) трактовать несуществующие прежде файлы как если бы они существовали, но имели пустое содержимое:

```
% diff -u -N newfile.orig newfile > patch-pathname-newfile
```

Файлы с патчами помещаются в каталоге PATCHDIR (как правило, это files/), откуда они будут взяты автоматически. Все патчи обязаны быть сделаны относительно каталога WRKSRC (как правило, это каталог, в который распаковывается исходный архив и где будет выполняться построение). Для упрощения внесения изменений и обновлений избегайте наличия более чем одного патча для одного и того же файла (например, патчей patch-file и patch-file2, оба меняющих файл WRKSRC/foobar.c). Обратите внимание, что если путь к изменяемому файлу содержит символ подчеркивания (_), то патч должен содержать в своем имени два подчеркивания вместо одного. Например, для применения патча на файл с именем src/freeglut_joystick.c соответствующий патч следует назвать patch-src-freeglut__joystick.c.

Пожалуйста, используйте для именования патчей только символы [-+._a-zA-Z0-9]. Не используйте любые другие символы, кроме этих. Не называйте патчи как patch-aa или patch-ab, всегда ссылайтесь на путь и название файла в названиях самих патчей.

Существует альтернативный упрощённый способ создания патчей для существующих файлов. Первые шаги те же самые: создание копии неизменённого файла с расширением .orig и внесение изменений. После этого используйте make makepatch, чтобы обновить файлы с патчами в каталоге files данного порта.

Не помещайте строки RCS в патчи. Subversion будет изменять их при помещении файлов в дерево портов, и когда мы будем их оттуда извлекать, они будут уже другие, поэтому применение патчей окончится неудачей. Строчки RCS предваряются знаком доллара ($), и обычно начинаются с $Id или $RCS.

Использование параметра рекурсии (-r) с командой diff(1) для генерации патчей - это хорошо, но всё же, пожалуйста, смотрите на получающиеся патчи, чтобы убедиться в отсутствии ненужного мусора. В частности, diff-разниц между двумя резервными копиями файлов, файлы Makefile, когда как порт использует Imake или GNU-версию программы configure, и так далее, не нужны, и должны быть удалены. Если было необходимо

отредактировать файл configure.in и запустить `autoconf` для перегенерации `configure`, не нужно включать файлы diff для `configure` (они частенько вырастают до нескольких тысяч строк!). Вместо этого задайте `USE_AUTOTOOLS=autoconf:261` и включите diff-файл для configure.in.

Старайтесь минимизировать в патчах объём нефункциональных изменений с пустыми символами. В мире Открытого Исходного Кода является распространенным совместное использование проектами больших объемов кодовой базы, но с различными стилями и правилами отступов. При копировании работающей функциональной части из одного проекта для исправления похожей области в другом, будьте аккуратны, пожалуйста: получаемый однострочный патч может указаться полон нефункциональных изменений. Это не только увеличивает размер репозитория Subversion, но также усложняет поиск того, что конкретно вызвало проблему и что вообще поменялось.

Если нужно удалить файл, сделайте это при выполнении цели `post-extract`, вместо того чтобы оформлять это как часть патча.

Простые перемещения могут быть выполнены непосредственно из Makefile порта с использованием sed(1) в режиме in-place. Это удобно, когда при изменении используется значение переменной:

```
post-patch:
    @${REINPLACE_CMD} -e 's|for Linux|for FreeBSD|g' ${WRKSRC}/README
```

Довольно часто в исходных файлах портируемого программного обеспечения используется конвенция CR/LF. Это может стать причиной проблем с дальнейшей упаковкой, предупреждениями компилятора или выполнением скриптов (таких как `/bin/sh^M not found`). Для быстрого преобразования всех файлов из CR/LF просто в LF добавьте в Makefile порта эту запись:

```
USES=   dos2unix
```

Может быть задан точный список преобразуемых файлов:

```
USES=   dos2unix
DOS2UNIX_FILES= util.c util.h
```

Используйте `DOS2UNIX_REGEX`, чтобы преобразовать группу файлов в разных подкаталогах. Его параметром является регулярное выражение, совместимое с find(1). Подробнее о формате в re_format(7). Такой вариант удобен для преобразования всех файлов заданного расширения. Для примера, преобразуем все исходные файлы, не затрагивая двоичные файлы:

```
USES=   dos2unix
DOS2UNIX_REGEX= .*\.([ch]|cpp)
```

Другим вариантом является использование `DOS2UNIX_GLOB`, который вызывает `find` для

каждого из перечисленных в нём элементов.

```
USES=   dos2unix
DOS2UNIX_GLOB=  *.c *.cpp *.h
```

4.5. Конфигурирование

Поместите все дополнительные команды, требуемые для настройки, в ваш скрипт configure и сохраните его в подкаталоге scripts. Как отмечено выше, вы можете сделать это целями в файле Makefile и/или скриптами с именами pre-configure или post-configure.

4.6. Обработка пользовательского ввода

Если для построения, конфигурации или установки вашего порта требуется некоторый ввод со стороны пользователя, то вы должны задать переменную `IS_INTERACTIVE` в вашем файле Makefile. В случае "ночного построения" это позволит пропустить ваш порт, если пользователь в своем окружении задал переменную `BATCH` (и если пользователь установил переменную `INTERACTIVE`, то будут строиться *только* порты, которые требуют взаимодействия с пользователем. Это сэкономит значительное количество времени на части машин, которые постоянно строят порты (смотрите ниже).

При наличии разумных ответов на задаваемые вопросы, подходящих по умолчанию, также рекомендуется проверять переменную `PACKAGE_BUILDING` и выключать интерактивный скрипт, если он есть. Это позволит нам строить пакеты для помещения на компакт-диски и FTP-серверы.

Глава 5. Настройка файла Makefile

Настройка файла Makefile достаточно проста, и мы снова предполагаем, что перед тем, как начать, вы посмотрите на существующие примеры. К тому же в этом руководстве имеется примерный Makefile, так что взгляните на него и, пожалуйста, следуйте порядку переменных и разделов в этом образце, чтобы облегчить чтение вашего порта другими людьми.

Итак, расположим решаемые задачи в порядке их возникновения при создании вашего нового файла Makefile:

5.1. Оригинальные исходный код

Находится ли он в каталоге `DISTDIR` в виде стандартного упакованного архиватором `gzip` tar-архива с именем типа foozolix-1.2.tar.gz? Если это так, можно перейти к следующему шагу. Если нет, то вы должны попытаться переопределить некоторые из переменных `DISTVERSION`, `DISTNAME`, `EXTRACT_CMD`, `EXTRACT_BEFORE_ARGS`, `EXTRACT_AFTER_ARGS`, `EXTRACT_SUFX` или `DISTFILES` в зависимости от того, насколько необычен формат дистрибутивного файла.

В худшем случае вы можете просто определить свою собственную цель `do-extract` для переопределения действий по умолчанию, хотя к этому нужно будет прибегать в очень редких случаях, если вообще придётся.

5.2. Именование

В первой части Makefile порта ему даётся название, указывается его номер версии и принадлежность к правильной категории.

5.2.1. `PORTNAME` и `PORTVERSION`

В переменной `PORTNAME` вы должны указать основную часть имени вашего порта, а в переменной `PORTVERSION` - номер версии.

5.2.2. `PORTREVISION` и `PORTEPOCH`

5.2.2.1. `PORTREVISION`

Переменная `PORTREVISION` представляет собой монотонно увеличивающееся число, которое обнуляется при каждом увеличении значения переменной `PORTVERSION` (то есть каждый раз, когда создателями выпускается новый официальный релиз), и добавляется к имени пакета, если оно не равно нулю. Изменения в `PORTREVISION` используются автоматизированными инструментами (например, `pkg version`, см. pkg-version(8)) для определения факта появления нового пакета.

Значение `PORTREVISION` должно увеличиваться каждый раз, когда в порте FreeBSD делаются изменения, которые как-либо меняют получаемый пакет. Сюда относятся только изменения, затрагивающие построение пакета с параметрами по умолчанию.

Примеры случаев, когда значение PORTREVISION должно быть увеличено:

- Добавление патчей для исправления уязвимостей, ошибок, или добавления новой функциональности в порт.
- Изменения в файле Makefile порта для включения и выключения параметров, определяемых при компиляции пакета.
- Изменения в списке упаковки или в поведении пакета во время его установки (например, изменение скрипта, генерирующего начальные данные для пакета, такие, как ssh-ключи для хоста).
- Увеличение версии динамической библиотеки, от которой зависит порт (в этом случае тот, кто попытается установить старый пакет после установки более новой версии библиотеки, не сможет этого сделать, потому что при этом будет делаться поиск старой библиотеки libfoo.x, а не libfoo.(x+1)).
- Большие функциональные изменения в дистрибутивном файле порта, происходящие без объявлений, и приводящие к большим изменениям, то есть изменения в дистрибутиве требуют корректировки файла distinfo без соответствующего изменения PORTVERSION, когда как команда `diff -ru` между новой и старой версиями показывает нетривиальные изменения в коде.

Примеры изменений, которые не требуют увеличения переменной PORTREVISION:

- Изменения стиля в скелете порта без функциональных изменений в пакете.
- Изменения в переменной MASTER_SITES или другие функциональные изменения порта, которые не затрагивают получающегося пакета.
- Тривиальные патчи к дистрибутивному файлу, такие, как исправления опечаток, которые не так уж важны, что пользователи пакета должны озаботиться обновлением.
- Исправления, касающиеся этапа построения, которые делают возможным построение пакета, если ранее это было невозможно сделать (пока изменения не приводят к изменению работы на любых других платформах, на которых порт ранее строился). Так как PORTREVISION отражает содержимое пакета, то, если ранее пакет не строился, то нет нужды увеличивать PORTREVISION для отметки изменения.

Правило, которому нужно приблизительно следовать, заключается в том, что нужно спрашивать себя, является ли вносимое в порт изменение таким, что от него выиграют все (в виде усовершенствования, исправления или благодаря тому, что новый пакет будет вообще работоспособным), и примите во внимание тот факт, что при этом все, кто регулярно обновляют своё дерево портов, будут обязаны это сделать. Если это так, то переменная PORTREVISION должна быть увеличена.

5.2.2.2. PORTEPOCH

Время от времени разработчик программного обеспечения или создатель порта FreeBSD делают что-то не так и выпускают версию программы, номер которой меньше предыдущей версии. Примером этого является порт, название которого меняется с foo-20000801 на foo-1.0 (изначально это не считалось бы более новой версией, так как 20000801 численно больше, чем 1).

Результат сравнения номера версии не всегда очевиден. Для выполнения сравнения двух строк с номером версии можно использовать `pkg version` (см. pkg-version(8)). Например:

```
% pkg version -t 0.031 0.29
>
```

Строка `>` в выводе команды означает, что версия 0.031 считается выше, чем версия 0.29, что может быть не очевидно для того, кто выполняет портирование.

В ситуациях, подобных этой, должно быть увеличено значение `PORTEPOCH`. Если значение `PORTEPOCH` не равно нулю, то оно добавляется к имени пакета, как описано в разделе выше. Значение `PORTEPOCH` никогда не должно уменьшаться или сбрасываться в ноль, потому что это приведёт к ошибке сравнения с пакетом с меньшим номером эпохи (то есть то, что пакет устарел, обнаружено не будет): номер новой версии (например, `1.0,1` в примере выше) останется меньше, чем номер предыдущей версии (20000801), однако суффикс `,1` интерпретируется различными автоматизированными утилитами особым образом, и окажется больше, чем предполагаемый суффикс `,0` более раннего пакета).

Некорректное уменьшение или сброс `PORTEPOCH` приводит к печальным последствиям; если вы не поняли, о чём шла речь ранее, пожалуйста, всё же разберитесь с этим, либо спросите в списках рассылки.

Предполагается, что в большинстве портов переменная `PORTEPOCH` использоваться не будет, но при корректном использовании `PORTVERSION` может появиться необходимость её иметь, если в будущих релизах программное обеспечение должно изменить структуру номера версии. Однако создателям портов для FreeBSD нужно быть внимательными, когда разработчик выпускает релиз без официального номера версии - эдакие "промежуточные" релизы. Имеется соблазн пометить релиз датой его выхода, что может вызвать проблемы, как и в примере выше, когда будет выпущен новый "официальный" релиз.

Например, если промежуточный релиз помечен датой 20000917, а предыдущая версия программного обеспечения имела номер 1.2, то промежуточному релизу должно быть поставлено в соответствие значение `PORTVERSION`, равное 1.2.20000917 или что-то похожее, но не 20000917, так как последующий релиз, скажем, 1.3, должен иметь численно большее значение.

5.2.2.3. Пример использования переменных `PORTREVISION` и `PORTEPOCH`

Выполнен коммит порта `gtkmumble`, версии `0.10`, в коллекцию портов.

```
PORTNAME=   gtkmumble
PORTVERSION=    0.10
```

Значение `PKGNAME` станет равным `gtkmumble-0.10`.

Обнаружена брешь в безопасности, исправление которой потребовало создания локального патча для FreeBSD. Соответственно было увеличено значение переменной PORTREVISION.

```
PORTNAME=   gtkmumble
PORTVERSION=    0.10
PORTREVISION=   1
```

PKGNAME принимает значение gtkmumble-0.10_1

Разработчиком выпущена новая версия с номером 0.2 (оказалось, что под номером 0.10 автор имел в виду 0.1.0, а не "то, что будет выпущено после версии 0.9" - извините, теперь уже поздно). Так как новый младший номер версии 2 по значению меньше, чем номер предыдущей версии 10, то должно быть увеличено значение PORTEPOCH для того, чтобы заставить распознавать вновь создаваемый пакет как "более новый". Так как это новый релиз программы, то PORTREVISION обнуляется (или удаляется из файла Makefile).

```
PORTNAME=   gtkmumble
PORTVERSION=    0.2
PORTEPOCH=  1
```

PKGNAME принимает значение gtkmumble-0.2,1

Следующий релиз имеет номер версии 0.3. Так как значение переменной PORTEPOCH никогда не уменьшается, что переменные, определяющие версии, теперь выглядят так:

```
PORTNAME=   gtkmumble
PORTVERSION=    0.3
PORTEPOCH=  1
```

PKGNAME принимает значение gtkmumble-0.3,1

Если значение PORTEPOCH этим обновлением было бы сброшено в 0, то кто-нибудь, имеющий установленный пакет gtkmumble-0.10_1, не смог бы опознать пакет gtkmumble-0.3 как более новый, так как 3 было бы меньше, чем 10. Помните, что в первую очередь это касается PORTEPOCH.

5.2.3. Переменные PKGNAMEPREFIX и PKGNAMESUFFIX

Две необязательные переменные, PKGNAMEPREFIX и PKGNAMESUFFIX, объединяются со значениями PORTNAME и PORTVERSION для формирования PKGNAME в форме ${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}. Добейтесь того, чтобы это соответствовало нашим рекомендациям по правильному выбору названий для пакетов. В частности, в переменной PORTVERSION *не разрешается* использование дефиса (-). Кроме того, если в имени пакета присутствует часть *language-* или *-compiled.specifics* (смотрите ниже), то используйте переменные PKGNAMEPREFIX и PKGNAMESUFFIX, соответственно. Не делайте их частью значения переменной PORTNAME.

5.2.4. Соглашения по именованию пакетов

Далее описаны некоторые соглашения, которым вы должны следовать в именовании ваших пакетов. Они были разработаны для облегчения просмотра каталога, так как имеется уже тысячи пакетов, а пользователи отвернутся от нас, если список не понравится их взору!

Имя пакета должно иметь вид language_region-name-compiled.specifics-version.numbers.

Имя пакета определяется как `${PKGNAMEPREFIX}${PORTNAME}${PKGNAMESUFFIX}-${PORTVERSION}`. Вы должны задавать значения переменных в соответствии с этим форматом.

1. FreeBSD пытается поддерживать языки, на которых разговаривают её пользователи. Часть *language-* должна быть двухсимвольным сокращением от названия языка по стандарту ISO-639, если порт специфичен для конкретного языка. Примерами являются `ja` для японского, `ru` для русского, `vi` для вьетнамского, `zh` для китайского, `ko` для корейского и `de` для немецкого языков.

   Если ваш порт специфичен для конкретного региона внутри области использования языка, добавьте также двухсимвольный код страны. Примерами являются `en_US` для US English и `fr_CH` для Swiss French.

   Часть *language-* должна задаваться в переменной `PKGNAMEPREFIX`.

2. Первая буква части name должна быть в нижнем регистре. (Оставшаяся часть названия может содержать буквы в верхнем регистре, так что принимайте решение сами, когда преобразуете имя программного пакета, содержащего в имени некоторое количество заглавных букв.) Существует традиция именовать модули для `Perl 5`, добавляя впереди `p5-` и преобразуя пару двоеточий в дефис; например, модуль `Data::Dumper` будет именоваться `p5-Data-Dumper`.

3. Убедитесь, что имя порта и версия четко отделены и размещаются в переменных `PORTNAME` и `PORTVERSION`. Единственная причина, по которой `PORTNAME` содержит версионную часть, это если полученный дистрибутив сам назван таким образом, как это сделано для портов textproc/libxml2 или japanese/kinput2-freewnn. В противном случае `PORTNAME` не должен содержать никакой информации, указывающей на версию. То, что некоторые порты имеют одинаковый `PORTNAME`, является вполне нормальным, как для портов www/apache*; в этом случае различные версии (и различные записи в индексе) отличаются по значениям `PKGNAMEPREFIX` и `PKGNAMESUFFIX`.

4. Если порт может быть построен с различными статически заданными значениями по умолчанию (обычно это часть имени каталога в семействе портов), то часть *-compiled.specifics* должна определять вкомпилированные значения по умолчанию (дефис не обязателен). Примерами являются размеры бумаги и шрифтов.

   Часть *-compiled.specifics* должна задаваться в переменной `PKGNAMESUFFIX`.

5. Строка с номером версии должна следовать за дефисом (`-`) и являться списком разделенных двоеточием чисел и букв в нижнем регистре. В частности, не разрешается иметь еще один дефис внутри строки с обозначением номера версии. Единственным исключением является строчка `pl` (означающая "patchlevel"), которая может использоваться *только* тогда, когда у программного обеспечения нет старшего и

младшего номера версии. Если в номер версии программного обеспечения включена строчка типа "alpha", "beta", "rc" или "pre", возьмите из неё первую букву и поставьте её непосредственно после точки. Если после таких строк номер версии ещё продолжается, то после буквы должно следовать число без дополнительной разделяющей точки.

Смысл такого формата заключается в удобстве сортировки портов по номеру версии. В частности, следите за тем, чтобы компоненты номера версии разделялись точкой, и если там присутствует дата, то используйте формат `0.0.yyyy.mm.dd`, но не `dd.mm.yyyy` или не совместимый с проблемой Y2K `yy.mm.dd`. Добавление к версии префикса `0.0.` является важным, в случае если выпущен релиз с присвоением настоящей версии, которая в числовом представлении, конечно же, будет ниже, чем `yyyy`.

Вот несколько (реальных) примеров того, как преобразовать имя из оригинального, придуманного авторами, к подходящему для имени пакета:

Имя дистрибутива	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	PORTVERSION	Обоснование
mule-2.2.2	(пусто)	mule	(пусто)	2.2.2	Изменений не потребовалось
EmiClock-1.0.2	(пусто)	emiclock	(пусто)	1.0.2	Для отдельных программ имена с заглавными буквами запрещены
rdist-1.3alpha	(пусто)	rdist	(пусто)	1.3.a	Строчки типа `alpha` запрещены
es-0.9-beta1	(пусто)	es	(пусто)	0.9.b1	Строчки типа `beta` запрещены
mailman-2.0rc3	(пусто)	mailman	(пусто)	2.0.r3	Строчки типа `rc` запрещены
v3.3beta021.src	(пусто)	tiff	(пусто)	3.3	Что это такое было вообще?
tvtwm	(пусто)	tvtwm	(пусто)	pl11	Всегда требуется указание номера версии

Имя дистрибутива	PKGNAMEPREFIX	PORTNAME	PKGNAMESUFFIX	PORTVERSION	Обоснование
piewm	(пусто)	piewm	(пусто)	1.0	Всегда требуется указание номера версии
xvgr-2.10pl1	(пусто)	xvgr	(пусто)	2.10.1	`pl` разрешено только при отсутствии старшего/младшего номера версии
gawk-2.15.6	ja-	gawk	(пусто)	2.15.6	Версия на японском языке
psutils-1.13	(пусто)	psutils	-letter	1.13	Размер бумаги задается статически во время построения пакета
pkfonts	(пусто)	pkfonts	300	1.0	пакет для шрифтов 300dpi

Если в исходном коде абсолютно нет информации о номере версии и не похоже, что автор собирается выпускать другую версию, то в качестве номера версии задайте просто `1.0` (как в примере с `piewm` выше). В противном случае спросите автора программы или используйте дату (`0.0.yyyy.mm.dd`) в качестве номера версии.

5.3. Разделение по категориям

5.3.1. `CATEGORIES`

В процессе создания пакета он помещается в каталог /usr/ports/packages/All, а в одном или более подкаталогов из /usr/ports/packages создаются на него ссылки. Имена этих подкаталогов определяются переменной `CATEGORIES`. Такая схема нужна для облегчения жизни пользователя, когда он сталкивается с массой пакетов на FTP-сервере или компакт-диске. Пожалуйста, посмотрите на текущий список категорий и выберите те из них, которые более всего подходят к вашему порту.

Этот список также определяет, куда в дереве портов будет помещен порт. Если вы укажете здесь более одной категории, то предполагается, что файлы порта будут помещены в

подкаталог с именем первой категории. Посмотрите ниже для получения подробной информации о том, как правильно выбрать категории.

5.3.2. Текущий список категорий

Вот текущий список категорий. Те, которые отмечены звёздочкой (*), являются *виртуальными* категориями-они не имеют собственного подкаталога в дереве портов. Они используются только в качестве вторичных категорий, и только для поиска.

Для невиртуальных категорий имеется однострочное описание в `COMMENT` в Makefile соответствующего подкаталога.

Категория	Описание	Примечания
accessibility	Порты для помощи пользователям с ограниченными возможностями.	
afterstep*	Порты, поддерживающие менеджер окон AfterStep.	
arabic	Поддержка арабского языка.	
archivers	Инструменты для работы с архивами.	
astro	Приложения, связанные с астрономией.	
audio	Поддержка работы со звуком.	
benchmarks	Утилиты для измерения производительности системы.	
biology	Программное обеспечение, связанное с биологией.	
cad	Инструменты Систем Автоматизированного Проектирования.	
chinese	Поддержка китайского языка.	
comms	Коммуникационное программное обеспечение.	В основном программы для работы с последовательным портом.
converters	Утилиты для преобразования символьных форматов.	
databases	Базы данных.	
deskutils	То, что было на столе до изобретения компьютеров.	

Категория	Описание	Примечания
devel	Утилиты для разработки программного обеспечения.	Не помещайте сюда библиотеки просто потому, что это библиотеки-если они подпадают под какую-то другую категорию, то их быть здесь не должно.
dns	Программное обеспечение для работы DNS.	
docs*	Мета-порты для документации FreeBSD.	
editors	Редакторы общего назначения.	Специализированные редакторы относят к разделу для соответствующих инструментов (например, редактор математических формул попадает в категорию math).
elisp*	Порты для Emacs lisp.	
emulators	Эмуляторы других операционных систем.	Эмуляторы терминалов сюда *не* относятся-те, которые разработаны для X, должны быть в категории x11, а текстовые в comms или misc, в зависимости от конкретного их предназначения.
finance	Приложения для работы с деньгами, финансами и всем, что с этим связано.	
french	Поддержка французского языка.	
ftp	Клиенты и серверы FTP.	Если ваш порт понимает как FTP, так и HTTP, поместите его в категорию ftp и укажите вторичную категорию www.
games	Игры.	
geography*	Программное обеспечение, связанное с географией.	
german	Поддержка немецкого языка.	
gnome*	Порты Проекта GNOME.	

Категория	Описание	Примечания
gnustep*	Программное обеспечение для окружения рабочего стола GNUstep.	
graphics	Графические утилиты.	
hamradio*	Программное обеспечение для любительского радио	
haskell*	Программное обеспечение, связанное с языком Haskell.	
hebrew	Поддержка иврита.	
hungarian	Поддержка венгерского языка.	
ipv6*	Программное обеспечение, связанное с IPv6.	
irc	Утилиты для Internet Relay Chat.	
japanese	Поддержка японского языка.	
java	Программное обеспечение, связанное с языком Java™.	Категория java ни в коем случае не должна быть единственной для порта. Оставьте для портов, непосредственно имеющих отношение к языку Java, портерам также рекомендуется не использовать java как основную категорию порта.
kde*	Порты проекта KDE.	
kld*	Загружаемые модули ядра.	
korean	Поддержка корейского языка.	
lang	Языки программирования.	
linux*	Linux приложения и утилиты.	
lisp*	Программное обеспечение, связанное с языком Lisp.	
mail	Программы для работы с почтой.	

Категория	Описание	Примечания
math	Программное обеспечение для численных вычислений и другие утилиты, связанные с математикой.	
mbone*	Приложения для MBone.	
misc	Различные утилиты	В общем, то, что не попадает в другие категории. Если это возможно, попробуйте найти более подходящую, чем `misc`, категорию для вашего порта, так как здесь порты теряются.
multimedia	Программное обеспечение для работы с мультимедиа.	
net	Различное сетевое программное обеспечение.	
net-im	Программы мгновенного обмена сообщениями.	
net-mgmt	Программное обеспечение для сетевого управления.	
net-p2p	Приложения для пиринговых сетей.	
news	Программное обеспечение для работы с конференциями USENET.	
palm	Программная поддержка Palm™.	
parallel*	Приложения, связанные с параллельными вычислениями.	
pear*	Порты, относящиеся к технологии Pear PHP.	
perl5*	Порты, которым для работы требуется Perl версии 5.	
plan9*	Различные программы из Plan9.	
polish	Поддержка польского языка.	
ports-mgmt	Порты для управления, установки и разработки портов и пакетов FreeBSD.	

Категория	Описание	Примечания
portuguese	Поддержка португальского языка.	
print	Программное обеспечение для печати.	Инструменты для вёрстки (просмотрщики и тому подобное) тоже относятся сюда.
python*	Программное обеспечение, связанное с языком Python.	
ruby*	Программное обеспечение, связанное с языком Ruby.	
rubygems*	Порты для пакетов RubyGems.	
russian	Поддержка русского языка.	
scheme*	Программное обеспечение, связанное с языком Scheme.	
science	Научные программы, которые не подпадают под другие категории, скажем, astro, biology или math.	
security	Программы, обеспечивающие безопасность системы.	
shells	Различные командные процессоры.	
sysutils	Системные утилиты.	
spanish*	Поддержка испанского языка.	
tcl*	Порты, для работы которых нужен Tcl.	
textproc	Утилиты для обработки текстов.	Инструменты для вёрстки помещаются в категорию print, а не сюда.
tk*	Порты, для работы которых нужен Tk.	
ukrainian	Поддержка украинского языка.	
vietnamese	Поддержка вьетнамского языка.	
windowmaker*	Порты для поддержки менеджера окон WindowMaker.	

Категория	Описание	Примечания
www	Программное обеспечение, связанное со всемирной паутиной.	Поддержка языка HTML относится сюда же.
x11	X Window System и иже с ними.	Эта категория предназначена только для программного обеспечения, которое поддерживает саму оконную систему. Не помещайте сюда обычные приложения для X: большинство из них должны быть перенесены в другие категории x11-* (смотрите ниже).
x11-clocks	Часы для X11.	
x11-drivers	Драйверы X11.	
x11-fm	Менеджеры файлов для X11.	
x11-fonts	Шрифты для X11 и утилиты для работы с ними.	
x11-servers	Серверы для X11.	
x11-themes	Темы для X11.	
x11-toolkits	Пакеты разработчика для X11.	
x11-wm	Оконные менеджеры для X11.	
xfce*	Порты, связанные с окружением рабочего стола Xfce.	
zope*	Поддержка Zope.	

5.3.3. Выбор правильной категории

Так как многие категории перекрываются, вам часто необходимо будет выбирать, какая их них должна быть основной для вашего порта. Есть несколько правил, по которым можно решить этот вопрос. Вот список приоритетов, в уменьшающейся степени предпочтения:

- Первая категория должна быть физической категорий (смотрите выше). Это необходимо для создания пакетов. После этого виртуальные и физические категории могут смешиваться.
- Сначала всегда идут категории, специфичные для языков. Например, если ваш порт устанавливает японские шрифты для X11, то строчка `CATEGORIES` должна иметь вид japanese x11-fonts.
- Более конкретные категории идут первыми перед более общими. В частности, редактор

HTML должен быть описан как www editors, а не наоборот. Кроме того, вы не должны указывать категорию net, если порт относится к одной из категорий irc, mail, news, security или www, так как net включается автоматически.

- x11 используется как вторичная категория только в случае, если в качестве основной категории указан естественный язык. В частности, вам не нужно указывать x11 в качестве категории для приложений X.
- Режимы для редактора Emacs должны помещаться в ту же категорию, что и приложение, которое поддерживается этим режимом, а не в editors. Например, режим Emacs для редактирования исходного кода некоторого языка программирования должен быть помещен в категорию lang.
- Порты, устанавливающие загружаемые модули ядра, должны содержать виртуальную категорию kld в строке `CATEGORIES`. Это одно из действий, выполняемых автоматически с добавлением `kmod` в строке `USES`.
- misc не должна указываться вместе с любой другой невиртуальной категорией. Если вы указываете `misc` вместе с чем-то ещё в строке `CATEGORIES`, это значит, что вы можете спокойно удалить `misc` и просто поместить порт в этот другой подкаталог!
- Если ваш порт решительным образом не подпадает ни под какую категорию, поместите его в misc.

Если вы не уверены в правильности выбора категории, пожалуйста, отметьте это в вашем сообщении send-pr(1), чтобы мы могли обсудить это до того, как включить порт в Коллекцию. Если вы являетесь коммиттером, пошлите замечание на адрес Список рассылки, посвящённый Портам FreeBSD, чтобы мы могли обсудить это. Зачастую новые порты помещаются не в ту категорию только для того, чтобы их оттуда сразу же удалили. Это приводит к излишнему и ненужному росту основного хранилища исходных текстов.

5.3.4. Предложение новой категории

Поскольку со временем Коллекция Портов увеличилась, то в связи с этим были добавлены различные новые категории. Новые категории могут быть или *виртуальными* категориями-которые не имеют соответствующего подкаталога в дереве портов-или *физическими* категориями-у которых он есть. Следующий текст содержит обсуждение вопросов, возникающих при создании новой физической категории, чтобы вы могли понимать их, когда предложите новую категорию.

В соответствие с существующей практикой мы избегаем создания новой физической категории, пока достаточно большое число портов логически ей не принадлежит или же порты, которые могли бы ей принадлежать, не являются логически обособленной группой, представляющей для всех ограниченный интерес (в частности, категории, относящиеся к естественным языкам); предпочтительно выполнение обоих условий.

Основной причиной для этого является то, что такое изменение создает изрядное количество работы и для коммиттеров, и для всех тех пользователей, которые отслеживают изменения в Коллекции Портов. В дополнение, предложенная категория создает естественное разногласие. (Пожалуй, потому что не существует четкого соглашения, является ли категория "слишком большой", или должны ли категории предоставлять себя для просмотра (и, таким образом, какое количество категорий было бы идеальным

значением), и так далее.)

Процедура:

1. Предложите новую категорию на Список рассылки, посвящённый Портам FreeBSD. Вам следует включить для новой категории детальное обоснование, в том числе почему вы считаете, что существующие категории не являются достаточными, и список существующих портов, предложенных для перемещения. (Если есть новые порты, ожидающие в GNATS и попадающие в эту категорию, то укажите их тоже.) Если вы являетесь сопровождающим и/или отправителем, то укажите это соответственно, так как это может помочь вам в вашем деле.
2. Принимайте участие в обсуждении.
3. Если кажется, что для вашей идеи появилась поддержка, отправьте PR, который будет включать обоснование и список существующих портов, которые надо переместить. В идеале этот PR должен также включать патчи для следующего:
   - Makefile'ы для новых портов в результате репозиторного копирования
   - Makefile для категорий старых портов
   - Makefile'ы для портов, зависящих от старых портов
   - (в дополнение, вы можете включить другие файлы, требующие изменений, согласно процедуре из Руководства Коммиттера.)
4. Поскольку это затрагивает инфраструктуру портов и охватывает не только выполнение репозиторного копирования, но также, возможно, и выполнение регрессивных тестов на кластере построения, то PR должна назначать себе Группа Менеджеров Дерева Портов FreeBSD <portmgr@FreeBSD.org>.
5. Если этот PR одобрен, то коммиттеру нужно продолжить остальную часть процедуры, которая изложена в Руководстве Коммиттера.

Предложение новой виртуальной категории должно быть схожим с вышеизложенным, но при этом затрагивать намного меньше, поскольку ни один из портов не будет перемещен в действительности. В этом случае единственными патчами, включенными в PR, будут те, что добавляют новую категорию в `CATEGORIES` каждого из затрагиваемых портов.

5.3.5. Предложение реорганизации всех категорий

Время от времени кто-нибудь предлагает произвести реорганизацию категорий либо до двухуровневой, либо другого типа на основе ключевых слов. На данный момент из этих предложений ничего не получилось, потому что, хотя они просты в реализации, но предполагаемая переделка всей коллекции портов по меньшей мере приводит в уныние. Пожалуйста, прочтите историю этих предложений в архивах рассылок перед тем, как присылать свои соображения; более того, вы должны быть готовы представить работающий прототип.

5.4. Дистрибутивные файлы

Во второй части Makefile задаётся, какие файлы и откуда должны быть сгружены для того, чтобы построить порт.

5.4.1. DISTVERSION/DISTNAME

В переменной DISTNAME указывается имя порта так, как назвали его создатели программного обеспечения. Значение DISTNAME по умолчанию совпадает с ${PORTNAME}-${PORTVERSION}, так что переопределяете её значение только в случае необходимости. DISTNAME используется только в двух местах. Во-первых, список дистрибутивных файлов (DISTFILES) по умолчанию состоит из ${DISTNAME}${EXTRACT_SUFX}. И во-вторых, предполагается, что дистрибутивный файл будет распакован в подкаталог с именем WRKSRC, значение которого по умолчанию есть не что иное, как work/${DISTNAME}.

Названия некоторых дистрибутивов, которые не укладываются в ${PORTNAME}-${PORTVERSION} -схему, могут быть автоматически обработаны посредством установки переменной DISTVERSION. PORTVERSION и DISTNAME будут унаследованы автоматически, но конечно же могут быть переопределены. Следующая таблица демонстрирует некоторые примеры:

DISTVERSION	PORTVERSION
0.7.1d	0.7.1.d
10Alpha3	10.a3
3Beta7-pre2	3.b7.p2
8:f_17	8f.17

Значения переменных PKGNAMEPREFIX и PKGNAMESUFFIX не влияют на значение DISTNAME. Заметьте также, что если значение WRKSRC равно work/${PORTNAME}-${PORTVERSION}, и в случае, когда оригинальный архив называется по имени, отличном от ${PORTNAME}-${PORTVERSION}${EXTRACT_SUFX}, скорее всего, вы должны оставить DISTNAME как есть- лучше переопределить DISTFILES, чем задавать значения как DISTNAME, так и WRKSRC (и, возможно, ещё и EXTRACT_SUFX).

5.4.2. MASTER_SITES

Содержит часть с каталогом FTP/HTTP-URL, которая указывает на оригинальный архив на сервере MASTER_SITES. Не забудьте лидирующий слэш (/)!

Макрос команды make будет пытаться воспользоваться этой переменной для получения дистрибутивного файла с помощью программы FETCH, если он не будет найден в системе.

Рекомендуется помещать в список много сайтов, предпочтительно с разных континентов. Это поможет при наличии проблем с мировой сетью. Мы даже планируем добавить поддержку автоматического определения ближайшего сайта и сгрузки файлов оттуда; наличие нескольких сайтов будет способствовать этому начинанию.

Если оригинальный архив находится на одном из таких популярных серверов, как SourceForge, GNU или Perl CPAN, то указывайте эти сайты в простой форме при помощи MASTER_SITE_* (к примеру, MASTER_SITE_SOURCEFORGE, MASTER_SITE_GNU или MASTER_SITE_PERL_CPAN. Просто укажите в переменной MASTER_SITES одно из этих значений, а в переменной MASTER_SITE_SUBDIR задайте путь к архиву. Вот пример:

```
MASTER_SITES=       ${MASTER_SITE_GNU}
MASTER_SITE_SUBDIR= make
```

Или можно использовать сокращенный формат:

```
MASTER_SITES=   GNU/make
```

Эти переменные определены в файле /usr/ports/Mk/bsd.sites.mk. Всё время добавляются новые записи, так что обращайтесь к последней версии этого файла перед тем, как послать нам свой порт.

Для популярных сайтов существует несколько *магических* макросов с заранее известной структурой каталогов. Используйте для них сокращения, и система попытается угадать для вас правильный подкаталог.

```
MASTER_SITES=   SF
```

Если попытка угадать не удалась, то это может быть переписано следующим образом.

```
MASTER_SITES=   SF/stardict/WyabdcRealPeopleTTS/${PORTVERSION}
```

Что также можно записать в таком виде:

```
MASTER_SITES=   SF
MASTER_SITE_SUBDIR= stardict/WyabdcRealPeopleTTS/${PORTVERSION}
```

Таблица 1. Популярные магические макросы для MASTER_SITES

Макрос	Предполагаемый подкаталог
APACHE_JAKARTA	/dist/jakarta/${PORTNAME:S,-,,/,}/source
BERLIOS	/${PORTNAME:L}
CHEESESHOP	/packages/source/source/${DISTNAME:C/(.).*/\1/}/${DISTNAME:C/(.*)-[0-9].*/\1/}
DEBIAN	/debian/pool/main/${PORTNAME:C/^((lib)?.).*$/\1/}/${PORTNAME}
GCC	/pub/gcc/releases/${DISTNAME}

Макрос	Предполагаемый подкаталог
GNOME	/pub/GNOME/sources/${PORTNAME}/${PORTVERSION:C/^(\.[0-9]).*/\1/}
GNU	/gnu/${PORTNAME}
MOZDEV	/pub/mozdev/${PORTNAME:L}
PERL_CPAN	/pub/CPAN/modules/by-module/${PORTNAME:C/-.*//}
PYTHON	/ftp/python/${PYTHON_PORTVERSION:C/rc[0-9]//}
RUBYFORGE	/${PORTNAME:L}
SAVANNAH	/${PORTNAME:L}
SF	/project/${PORTNAME:L}/${PORTNAME:L}/${PORTVERSION}

5.4.3. EXTRACT_SUFX

Если у вас имеется один дистрибутивный файл, и в его имени используется странное окончание для указания типа сжатия, задайте переменную EXTRACT_SUFX.

К примеру, если дистрибутивный файл носит имя foo.tgz, а не более привычное foo.tar.gz, вы должны написать:

```
DISTNAME=   foo
EXTRACT_SUFX=   .tgz
```

Переменные USE_BZIP2, USE_XZ и USE_ZIP при необходимости автоматически устанавливают значение EXTRACT_SUFX в .tar.bz2, .tar.xz или .zip. Если ни одна из этих переменных не задана, то значение EXTRACT_SUFX по умолчанию устанавливается в .tar.gz.

Вам не нужно задавать значения EXTRACT_SUFX и DISTFILES одновременно.

5.4.4. DISTFILES

Иногда имена сгружаемых файлов не соответствуют имени порта. К примеру, файл может называться source.tar.gz или подобным образом. В других случаях исходный код приложения может располагаться в нескольких отличающихся архивах, и все они должны быть сгружены.

Если это ваш случай, то задайте в переменной DISTFILES список разделённых пробелами имён файлов, которые нужно сгрузить.

```
DISTFILES=  source1.tar.gz source2.tar.gz
```

Если переменная DISTFILES не задана явно, то её значением по умолчанию будет ${DISTNAME}${EXTRACT_SUFX}.

5.4.5. EXTRACT_ONLY

Если только некоторые из DISTFILES должны быть распакованы-к примеру, часть из них является исходным кодом, а другие представляют собой неупакованную документацию-перечислите имена файлов, которые должны быть распакованы, в EXTRACT_ONLY.

```
DISTFILES=  source.tar.gz manual.html
EXTRACT_ONLY=   source.tar.gz
```

Если *ни один* из DISTFILES не должен распаковываться, то установите пустое значение переменной EXTRACT_ONLY.

```
EXTRACT_ONLY=
```

5.4.6. PATCHFILES

Если вашему порту требуются некоторых дополнительные патчи, которые доступны по FTP или HTTP, задайте имена этих файлов в переменной PATCHFILES, а в переменной PATCH_SITES укажите URL того каталога, в котором они содержатся (формат такой же, как для MASTER_SITES).

Если патч не относится к самому верху дерева исходных текстов (то есть WRKSRC), потому что он содержит некоторые дополнительные пути, установите соответственно значение переменной PATCH_DIST_STRIP. В частности, если все имена путей в патче имеют дополнительный путь foozolix-1.0/ перед именем файла, то задайте PATCH_DIST_STRIP=-p1.

Не волнуйтесь, если патчи упакованы; они будут распакованы автоматически, если имена файлов оканчиваются на .gz или .Z.

Если патч распространяется вместе с какими-то другими файлами, такими, как документация, в виде tar-архива gzip, вы не можете просто использовать PATCHFILES. Если это ваш случай, добавьте имя и местоположение архива с патчем к DISTFILES и MASTER_SITES. Затем воспользуйтесь переменной EXTRA_PATCHES для указания этих файлов, и bsd.port.mk автоматически применит эти патчи. В частности, *не копируйте* файлы с патчами в каталог PATCHDIR-этот каталог может быть недоступным для записи.

Архив будет распакован вне исходного кода, как обычно, и к тому же его не нужно явно распаковывать, если это обычный архив gzip или compress. Если вы сделаете последнее, приложите дополнительные усилия для того, чтобы не перезаписать что-либо, уже существующее в этом каталоге. Также не забудьте добавить команду для удаления скопированного патча в цели pre-clean.

5.4.7. Несколько дистрибутивных файлов или патчей с различных серверов и подкаталогов (MASTER_SITES:n)

(Этот раздел можно считать немного "повышенной трудности"; те, кто впервые знакомятся с этим текстом, могут пропустить этот раздел).

В этом разделе находится информация о механизме сгрузки, известном как MASTER_SITES:n и MASTER_SITES_NN. Далее мы будем называть этот механизм MASTER_SITES:n.

Сначала немного общей информации. В OpenBSD имеется полезная возможность, используемая в переменных DISTFILES и PATCHFILES, которая позволяет закреплять после имен файлов и патчей идентификаторы типа :n. Здесь n может быть из диапазона [0-9] и обозначать закреплённую группу. К примеру:

```
DISTFILES=  alpha:0 beta:1
```

В OpenBSD дистрибутивный файл alpha будет связан с переменной MASTER_SITES0, но не с нашей общей переменной MASTER_SITES, а файл beta с переменной MASTER_SITES1.

Этот очень интересная возможность, которая может уменьшить этот бесконечный поиск работающего сайта для сгрузки.

Просто представьте себе 2 файла в DISTFILES и 20 сайтов в MASTER_SITES; сайты очень медленные, причём beta находится на всех сайтах из MASTER_SITES, а alpha может быть найден только на 20-м сайте. Будет неправильно проверять их все, если создатель знает об этом, не правда ли? Неподходящее начало для таких прекрасных выходных!

Теперь, когда вы получили общее представление, просто представьте ещё большее количество DISTFILES и MASTER_SITES. Конечно, наш "магистр доступности дистрибутивов" представляет масштабы нагрузки на сеть, которую это даёт.

В последующих разделах информация будет даваться вместе с реализацией этой идеи во FreeBSD. Мы несколько улучшили концепцию OpenBSD.

5.4.7.1. Упрощённая информация

В этом разделе рассказывается, как быстро подготовить точную сгрузку нескольких дистрибутивных файлов и патчей с разных сайтов и каталогов. Мы описываем здесь случай упрощённого использования MASTER_SITES:n. Для большинства сценариев этого будет достаточно. Однако, если вам нужна дополнительная информация, обратитесь к следующему разделу.

Некоторые приложения состоят из многих дистрибутивных файлов, которые должны быть сгружены с нескольких различных сайтов. К примеру, Ghostscript состоит из основной программы и большого числа файлов драйверов, которые используются в зависимости от принтера пользователя. Некоторые из этих файлов драйверов поставляются с основной программой, но при этом многие другие должны быть сгружены с множества различных сайтов.

Чтобы это поддерживать, за каждой записью в DISTFILES может следовать символ двоеточия и "имя метки". За каждым сайтом, перечисленным в MASTER_SITES, тоже следует двоеточие и метка, которая указывает, какие файлы дистрибутива должны быть сгружены с этого сайта.

Например, рассмотрим приложение, исходный код которого разделён на две части, source1.tar.gz и source2.tar.gz, которые должны быть сгружены с двух различных источников. Файл Makefile порта будет содержать строчки типа Упрощённое использование MASTER_SITES:n с 1 файлом на каждом сайте.

Пример 1. Упрощённое использование MASTER_SITES:n с 1 файлом на каждом сайте

```
MASTER_SITES=   ftp://ftp.example1.com/:source1 \
        ftp://ftp.example2.com/:source2
DISTFILES=  source1.tar.gz:source1 \
        source2.tar.gz:source2
```

Несколько дистрибутивных файлов могут иметь одну и ту же метку. Продолжая предыдущий пример, положим, что имеется и третий дистрибутивный файл, source3.tar.gz, который должен быть сгружен с ftp.example2.com. Тогда файл Makefile будет написан как Упрощённое использование MASTER_SITES:n с более чем 1 файлом на каждом сервере.

Пример 2. Упрощённое использование MASTER_SITES:n с более чем 1 файлом на каждом сервере

```
MASTER_SITES=   ftp://ftp.example1.com/:source1 \
        ftp://ftp.example2.com/:source2
DISTFILES=  source1.tar.gz:source1 \
        source2.tar.gz:source2 \
        source3.tar.gz:source2
```

5.4.7.2. Подробная информация

Прекрасно, но пример из предыдущего раздела не показал вам всё, что вам нужно? В этом разделе мы подробно опишем, как работает механизм MASTER_SITES:n точной сгрузки и как вы можете изменить ваши порты, чтобы это использовать.

1. За элементами могут следовать символы :n, где *n* это `, то есть _n_ может теоретически быть любой алфавитно-цифровой строкой, но пока мы будем ограничивать их `[a-zA-Z_][0-9a-zA-Z_].

   Более того, совпадение строк чувствительно к регистру; другими словами, n отличается от N.

   Однако следующие слова не могут использоваться для этих нужд, так как они имеют особое значение: default, all и ALL (они используются для своих нужд в ii). Кроме того, DEFAULT является специальным ключевым словом (посмотрите 3).

2. Элементы, за которыми следуют `:n`, принадлежат группе `n`, `:m` относится к группе `m` и так далее.
3. Элементы без таких суффиксов не относятся ни к какой группе, то есть они принадлежат к особой группе `DEFAULT`. Если вы укажете суффиксом любого элемента `DEFAULT`, вы просто выполните излишнюю работу, если только вы не хотите отнесения элемента как к группе `DEFAULT`, так и какой-то другой в одно и то же время (посмотрите на пункт 5).

   Следующие примеры равнозначны, но первый более предпочтителен:

   ```
   MASTER_SITES=   alpha
   ```

   ```
   MASTER_SITES=   alpha:DEFAULT
   ```

4. Группы не являются эксклюзивными, элемент может принадлежать к нескольким отличающимся группам одновременно, а группа может либо иметь несколько различных элементов, либо не иметь их вовсе. Повторяющиеся элементы в одной и той же группе будут являться просто повторяющимися элементами.
5. Если в хотите, чтобы элемент принадлежал к нескольким группам одновременно, вы можете использовать запятую (`,`).

   Вместо того, чтобы повторять их несколько раз, каждый раз с разным постфиксом, мы можем перечислить несколько групп за раз в одном постфиксе. Например, `:m,n,o` определяет элемент, принадлежащий группам `m`, `n` и `o`.

   Все следующие примеры имеют один смысл, но последний является предпочтительным:

   ```
   MASTER_SITES=   alpha alpha:SOME_SITE
   ```

   ```
   MASTER_SITES=   alpha:DEFAULT alpha:SOME_SITE
   ```

   ```
   MASTER_SITES=   alpha:SOME_SITE,DEFAULT
   ```

   ```
   MASTER_SITES=   alpha:DEFAULT,SOME_SITE
   ```

6. Все серверы внутри определённой группы сортируются в соответствии с `MASTER_SORT_AWK`. Все группы в `MASTER_SITES` и `PATCH_SITES` тоже сортируются.
7. Семантика групп может использоваться в любой из следующих переменных `MASTER_SITES`, `PATCH_SITES`, `MASTER_SITE_SUBDIR`, `PATCH_SITE_SUBDIR`, `DISTFILES` и `PATCHFILES` в соответствии со следующим синтаксисом:

   a. Все элементы `MASTER_SITES`, `PATCH_SITES`, `MASTER_SITE_SUBDIR` и `PATCH_SITE_SUBDIR` должны заканчиваться символом прямого слэша `/`. Если какие-то элементы принадлежат

каким-то группам, постфикс группы `:n` должен следовать сразу после завершающего символа `/`. Механизм `MASTER_SITES:n` опирается на наличие завершающего символа `/` во избежание совпадающих элементов, где `:n` является корректной частью элемента с вхождениями, где `:n` обозначает группу `n`. Для целей совместимости, так как завершающий символ `/` ранее не требовался в элементах `MASTER_SITE_SUBDIR` и `PATCH_SITE_SUBDIR`, если символ, сразу предшествующий постфиксу, не является символом `/`, то `:n` будет считаться корректной частью элемента, а не постфиксом группы, даже если за элементом следует `:n`. Посмотрите Подробное использование `MASTER_SITES:n` в `MASTER_SITE_SUBDIR` и Подробное использование `MASTER_SITES:n` с запятыми, несколькими файлами, несколькими серверами и несколькими подкаталогами.

Пример 3. Подробное использование `MASTER_SITES:n` в `MASTER_SITE_SUBDIR`

```
MASTER_SITE_SUBDIR= old:n new/:NEW
```

- Каталоги внутри группы `DEFAULT` → old:n
- Каталоги внутри группы `NEW` → new

Пример 4. Подробное использование `MASTER_SITES:n` с запятыми, несколькими файлами, несколькими серверами и несколькими подкаталогами

```
MASTER_SITES=   http://site1/%SUBDIR%/ http://site2/:DEFAULT \
        http://site3/:group3 http://site4/:group4 \
        http://site5/:group5 http://site6/:group6 \
        http://site7/:DEFAULT,group6 \
        http://site8/%SUBDIR%/:group6,group7 \
        http://site9/:group8
DISTFILES=  file1 file2:DEFAULT file3:group3 \
        file4:group4,group5,group6 file5:grouping \
        file6:group7
MASTER_SITE_SUBDIR= directory-trial:1 directory-n/:groupn \
            directory-one/:group6,DEFAULT \
            directory
```

Предыдущий пример приводит к следующей точной сгрузке. Серверы перечислены в точном порядке их использования.

- file1 будет сгружаться с
  - `MASTER_SITE_OVERRIDE`
  - http://site1/directory-trial:1/
  - http://site1/directory-one/
  - http://site1/directory/
  - http://site2/

    - http://site7/
    - MASTER_SITE_BACKUP
  - file2 будет сгружаться точно также, как file1, так как они оба относятся к одной и той же группе
    - MASTER_SITE_OVERRIDE
    - http://site1/directory-trial:1/
    - http://site1/directory-one/
    - http://site1/directory/
    - http://site2/
    - http://site7/
    - MASTER_SITE_BACKUP
  - file3 будет сгружен с
    - MASTER_SITE_OVERRIDE
    - http://site3/
    - MASTER_SITE_BACKUP
  - file4 будет сгружаться с
    - MASTER_SITE_OVERRIDE
    - http://site4/
    - http://site5/
    - http://site6/
    - http://site7/
    - http://site8/directory-one/
    - MASTER_SITE_BACKUP
  - file5 будет сгружен с
    - MASTER_SITE_OVERRIDE
    - MASTER_SITE_BACKUP
  - file6 будет сгружаться с
    - MASTER_SITE_OVERRIDE
    - http://site8/
    - MASTER_SITE_BACKUP

8. Как мне группировать одну из специальных переменных из bsd.sites.mk, например, MASTER_SITE_SOURCEFORGE?

   Посмотрите Подробное использование MASTER_SITES:n с MASTER_SITE_SOURCEFORGE.

Пример 5. Подробное использование `MASTER_SITES:n` *с* `MASTER_SITE_SOURCEFORGE`

```
MASTER_SITES=   http://site1/ ${MASTER_SITE_SOURCEFORGE:S/$/:sourceforge,TEST/}
DISTFILES=  something.tar.gz:sourceforge
```

something.tar.gz будет сгружаться со всех сайтов из `MASTER_SITE_SOURCEFORGE`.

9. Как мне использовать это с переменными `PATCH*`?

   Все примеры выполнялись с переменными `MASTER*`, и они работают точно так же и для `PATCH*`, как это можно видеть в Упрощённое использование `MASTER_SITES:n` с `PATCH_SITES`..

   Пример 6. Упрощённое использование `MASTER_SITES:n` *с* `PATCH_SITES`.

```
PATCH_SITES=    http://site1/ http://site2/:test
PATCHFILES= patch1:test
```

5.4.7.3. Что изменится для портов? А что не изменится?

i. Все имеющиеся порты остаются без изменений. Код для механизма `MASTER_SITES:n` активируется, если только есть элементы, которые заканчиваются на `:n`, как и элементы в соответствии с вышеописанным синтаксисом, особенно как это показано в пункте 7.

ii. Цели порт остаются теми же самыми: `checksum`, `makesum`, `patch`, `configure`, `build` и так далее. С обычными исключениями для `do-fetch`, `fetch-list`, `master-sites` и `patch-sites`.

   - `do-fetch`: использует новую группировку с постфиксами в `DISTFILES` и `PATCHFILES` с соответствующими элементами групп в `MASTER_SITES` и `PATCH_SITES`, которые используют группы из `MASTER_SITE_SUBDIR` и `PATCH_SITE_SUBDIR`. Посмотрите Подробное использование `MASTER_SITES:n` с запятыми, несколькими файлами, несколькими серверами и несколькими подкаталогами.
   - `fetch-list`: работает так же, как старая цель `fetch-list` с тем исключением, что она группирует, как и `do-fetch`.
   - `master-sites` и `patch-sites`: (несовместимы со старыми версиями) только возвращают элементы группы `DEFAULT`; на самом деле они выполняют цели `master-sites-default` и `patch-sites-default` соответственно.

     Более того, использование целей `master-sites-all` или `patch-sites-all` предпочтительно для непосредственной проверки `MASTER_SITES` или `PATCH_SITES`. Также работа прямой проверки в последующих версиях не гарантируется. Посмотрите B для получения более дополнительной информации об этих новых целях.

iii. Новые цели построения портов

   a. Имеются цели `master-sites-n` и `patch-sites-n`, которые будут перечислять элементы соответствующей группы *n* из `MASTER_SITES` и `PATCH_SITES` соответственно. К примеру,

master-sites-DEFAULT и patch-sites-DEFAULT обе будут возвращать элементы группы DEFAULT, master-sites-test и patch-sites-test группы test и так далее.

b. Имеются новые цели master-sites-all и patch-sites-all, которые выполняют работу старых master-sites и patch-sites. Они возвращают элементы всех групп, как если бы они все принадлежали одной и той же группе с тем, что она перечисляет ровно столько MASTER_SITE_BACKUP и MASTER_SITE_OVERRIDE, как и группы, определённые в DISTFILES или PATCHFILES; соответственно для master-sites-all и patch-sites-all.

5.4.8. DIST_SUBDIR

Не позволяйте вашему порту засорять /usr/ports/distfiles. Если вашему порту требуется сгрузить много файлов, или он содержит имя файла, могущее вызвать конфликты с другими портами (например, Makefile), то укажите в переменной DIST_SUBDIR имя порта (должны подойти ${PORTNAME} или ${PKGNAMEPREFIX}${PORTNAME}). Это изменит значение переменной DISTDIR со значения по умолчанию /usr/ports/distfiles к значению /usr/ports/distfiles/DIST_SUBDIR, и в результате всё, что требуется для порта, будет помещено в этот подкаталог.

Он заглянет также в подкаталог с тем же именем на основном резервном сервере ftp.FreeBSD.org. (Явное задание переменной DISTDIR в вашем файле Makefile этого не сделает, так что, пожалуйста, воспользуйтесь DIST_SUBDIR.)

Это не коснётся тех сайтов MASTER_SITES, которые вы указали в вашем файле Makefile.

5.4.9. ALWAYS_KEEP_DISTFILES

Если ваш порт использует двоичные дистрибутивные файлы и обладает лицензией, требующей, чтобы исходный код предоставлялся вместе с пакетами, распространяемыми в двоичной форме, например GPL, то ALWAYS_KEEP_DISTFILES даст кластеру построения FreeBSD указание сохранять копию файлов, указанных в DISTFILES. Пользователям таких портов эти файлы в основном не нужны, поэтому хорошей идеей является добавление в DISTFILES исходных дистрибутивных файлов, только когда определена переменная PACKAGE_BUILDING.

Пример 7. Использование ALWAYS_KEEP_DISTFILES.

```
.if defined(PACKAGE_BUILDING)
DISTFILES+=     foo.tar.gz
ALWAYS_KEEP_DISTFILES=  yes
.endif
```

При добавлении дополнительных файлов в DISTFILES убедитесь, что вы их также добавляете в distinfo. Кроме того, дополнительные файлы обычно распаковываются также в WRKDIR, что для некоторых портов может вызывать нежелательные подобные эффекты и требовать особую обработку.

5.5. MAINTAINER

Укажите здесь ваш адрес электронной почты. Пожалуйста. *:-)*

Заметьте, что в качестве значения для `MAINTAINER` допустимо использование только одного адреса без поля комментария. Должен использоваться формат `user@hostname.domain`. Пожалуйста, не включайте никакого описательного текста, например, вашего настоящего имени в эту строку-это несколько сбивает с толку bsd.port.mk.

Сопровождающий ответственен за поддержание порта в актуальном состоянии и обеспечение правильной работы порта. За подробным описанием обязанностей сопровождающего порт обращайтесь к главе The challenge for port maintainers.

Перед фиксацией в репозитории изменения в порте будут отправлены сопровождающему для просмотра и одобрения. Если сопровождающий порта не ответил на запрос пользователя об обновлении в течение двух недель (исключая большие праздники), то это можно считать тайм-аутом сопровождающего, и обновление может быть выполнено без явного подтверждения от сопровождающего. Если сопровождающий не отвечает в течение трёх месяцев, то считается, что он отсутствует, и как сопровождающий порта, о котором идёт речь, может быть заменён. Исключениями из этого правила является всё, что сопровождает Группа Менеджеров Дерева Портов FreeBSD <portmgr@FreeBSD.org> или Группа Офицеров Безопасности <security-officer@FreeBSD.org>. Запрещено делать любые несанкционированные изменения в портах, которые ведут эти группы.

Мы оставляем за собой право изменять сообщение сопровождающего для лучшего соответствия существующим политикам и стилю Коллекции Портов без явного одобрения со стороны отправителя. Также, крупные изменения в инфраструктуре могут повлечь изменения в порте без согласия сопровождающего. Такой вид изменений никогда не будет затрагивать функциональность порта.

За Группа Менеджеров Дерева Портов FreeBSD <portmgr@FreeBSD.org> оставляется право снять или назначить кого-либо сопровождающим по любой причине, а за Группа Офицеров Безопасности <security-officer@FreeBSD.org> оставляется право лишать или назначать права на сопровождение порта по соображениям информационной безопасности.

5.6. COMMENT

Содержит однострочное описание порта. Пожалуйста, соблюдайте следующие правила:

1. Старайтесь делать строку COMMENT длиной не больше, чем 70 символов, так как эта строка будет использована командой `pkg info` (см. pkg-info(8)) для отображения однострочного описания порта;
2. *Не* включайте сюда название пакета (или номер версии программного обеспечения);
3. Комментарий должен начинаться с заглавной буквы и не заканчиваться точкой;
4. Не начинайте комментарий с неопределённого артикля (A или An);
5. Имена пишутся с заглавной буквы (например, Apache, JavaScript, Perl);
6. Для перечислений используйте английскую Оксфордскую запятую (англ. Oxford comma)

(например, green, red, and blue);

7. Используйте программу проверки орфографии.

Вот пример:

```
COMMENT=    Cat chasing a mouse all over the screen
```

В файле Makefile переменная COMMENT должна следовать сразу за переменной MAINTAINER.

5.7. PORTSCOUT

Portscout являет собой автоматизированное средство проверки доступности дистрибутивных файлов для Коллекции Портов FreeBSD, подробное описание которого предоставляет Portscout: сканер дистрибутивных файлов портов FreeBSD.

Переменная PORTSCOUT задаёт специальные условия, ограничивающие работу Portscout - сканера дистрибутивных файлов.

Ситуации, при которых следует указывать переменную PORTSCOUT:

- Когда должны игнорироваться дистрибутивные файлы для конкретных версий или младших ревизий. Например, чтобы исключить из проверок новых версий дистрибутивных файлов версию *8.2* по причине того, что она является поломанной, добавьте следующее:

  ```
  PORTSCOUT=  ignore:8.2
  ```

- Когда должны проверяться конкретные версии или старшие и младшие ревизии дистрибутивных файлов. Например, если следует ограничиться проверкой версии *0.6.4*, потому что более новые версии имеют проблемы совместимости с FreeBSD, добавьте:

  ```
  PORTSCOUT=  limit:^0\.6\.4
  ```

- Когда URL, в которых указаны доступные версии, отличаются от URL их загрузки. Например, чтобы привязать проверку новых версий дистрибутивных файлов к странице загрузки для порта databases/pgtune, добавьте:

  ```
  PORTSCOUT=  site:http://pgfoundry.org/frs/?group_id=1000416
  ```

5.8. Зависимости

Многие порты зависят от других портов. Это очень удобная замечательная особенность большинства Unix-подобных операционных систем, включая FreeBSD. Множество портов

могут использовать общую зависимость совместно, а не включать её в состав каждого порта или пакета, который в ней нуждается. Имеется семь переменных, которые вы можете использовать для обеспечения того, что всё требуемое находится на машине пользователя. Имеется также несколько предопределённых переменных, отражающих зависимости для общих случаев, плюс ещё несколько для управления поведением зависимостей.

5.8.1. LIB_DEPENDS

Эта переменная указывает, от каких совместно используемых библиотек зависит порт. Это список пар lib:dir, где *lib* - это имя библиотеки, *dir* - это каталог, в котором можно ее найти в случае, если ее нет на машине. Например,

```
LIB_DEPENDS=    libjpeg.so:${PORTSDIR}/graphics/jpeg
```

проверит наличие библиотеки jpeg с любым номером версии и перейдет в подкаталог graphics/jpeg вашего дерева портов для ее построения и установки, если библиотека отсутствует.

Зависимость проверяется дважды, один раз внутри цели `build`, а затем из цели `install`. Кроме того, имя зависимости помещается в пакет, так что `pkg install` (см. pkg-install(8)) будет автоматически её устанавливать, если её нет на пользовательской системе.

5.8.2. RUN_DEPENDS

В этой переменной перечисляются выполнимые файлы или файлы, от которых зависит работа порта. Это список пар вида path:dir:target, где *path* - это имя программы или файла, а *dir* - каталог, в котором можно найти порт в случае, если его нет в системе, и *target* - это цель, которую нужно вызвать в этом каталоге. Если *path* начинается со слэша (/), он воспринимается как файл и его существование проверяется командой `test -e`; в противном случае предполагается, что это выполнимый файл, и для определения того, имеется ли программа в пути поиска, используется команда `which -s`.

Например,

```
RUN_DEPENDS=    ${LOCALBASE}/news/bin/innd:${PORTSDIR}/news/inn \
        xmlcatmgr:${PORTSDIR}/textproc/xmlcatmgr
```

проверит существование файла или каталога /usr/local/news/bin/innd, и если ничего не будет найдено, то построит и установит порт из подкаталога news/inn дерева портов. Также будет выполнена проверка, присутствует ли в пути поиска исполняемый файл с именем `xmlcatmgr`, и перейдет в подкаталог textproc/xmlcatmgr вашего дерева портов для его построения и установки, если он не будет найден.

В приведенном примере `innd` является выполнимым файлом; если выполнимый файл находится в месте, которое отсутствует в списке путей файлов, то вы должны указать полный путь к файлу.

Официальным значением переменной поиска PATH, используемым в кластере построения портов является

```
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin
```

Зависимость проверяется внутри цели install. Кроме того, имя зависимости помещается в пакет, так что pkg install (см. pkg-install(8)) будет автоматически его устанавливать, если он не будет найден в пользовательской системе. Часть *target* может быть опущена, если она совпадает с DEPENDS_TARGET.

Довольно распространенной является ситуация, когда RUN_DEPENDS буквально такая же как BUILD_DEPENDS, особенно если переносимое программное обеспечение написано на языке сценариев, или если оно требует такое же окружение для исполнения, как и используемое во время построения. В этом случае, очень заманчивым или довольно естественным является присвоение одного другому:

```
RUN_DEPENDS=    ${BUILD_DEPENDS}
```

Тем не менее, подобные присвоения могут загрязнять зависимости времени исполнения содержимым, не заданным в BUILD_DEPENDS исходного порта. Такое случается из-за ленивого вычисления в make(1) присваиваемых переменных. Представьте Makefile с переменными USE_*, которые обрабатываются в ports/Mk/bsd.*.mk для пополнения первоначальных зависимостей построения. Например, USES= gmake добавляет devel/gmake в BUILD_DEPENDS. Для предотвращения загрязнения RUN_DEPENDS подобными дополнительными зависимостями проявляйте осторожность с присвоением с раскрытием, т.е. с раскрытием значения перед его присвоением переменной:

```
RUN_DEPENDS:=   ${BUILD_DEPENDS}
```

5.8.3. BUILD_DEPENDS

В этой переменной перечисляются выполнимые или обычные файлы, которые требуются порту для его построения. Как и RUN_DEPENDS, это список пар path:dir:target. Например,

```
BUILD_DEPENDS=  unzip:${PORTSDIR}/archivers/unzip
```

будет проверять наличие выполнимого фала с именем unzip и перейдет в подкаталог archivers/unzip вашего дерева портов для его построения и установки, если последний не будет найден.

Под "построением" здесь понимается всё, от распаковки до компиляции. Зависимость проверяется из цели extract. Часть *target* может быть опущена, если она совпадает с DEPENDS_TARGET.

5.8.4. FETCH_DEPENDS

В этой переменной перечисляются выполняемые файлы или просто файлы, которые требуются порту для сгрузки. Как и предыдущие две переменные, это список пар path:dir:target. Например,

```
FETCH_DEPENDS=  ncftp2:${PORTSDIR}/net/ncftp2
```

будет проверять наличие выполняемого файла с именем `ncftp2` и перейдет в каталог net/ncftp2 вашего дерева портов для его построения и установки, если тот не будет найден.

Зависимость проверяется при выполнении цели `fetch`. Часть *target* может быть опущена, если она совпадает с `DEPENDS_TARGET`.

5.8.5. EXTRACT_DEPENDS

В этой переменной указываются программы или файлы, которые требуются для распаковки порта. Как и в предыдущих случаях, это список пар вида path:dir:target. Например,

```
EXTRACT_DEPENDS=    unzip:${PORTSDIR}/archivers/unzip
```

будет проверять наличие программы с именем `unzip`, и перейдёт в подкаталог archivers/unzip вашего дерева портов для её построения и установки, если такой программы не будет найдено.

Зависимость проверяется внутри цели `extract`. Часть *target* может быть опущена, если она совпадает с `DEPENDS_TARGET`.

Используйте эту переменную, только если распаковка не работает (по умолчанию предполагается использование `gzip`) и это не исправляется при помощи `USE_ZIP` или `USE_BZIP2`, которые описаны в `USE_*`.

5.8.6. PATCH_DEPENDS

Эта переменная указывает на программы или файлы, которые нужны порту для применения патчей. Как и в предыдущих случаях, это список пар вида path:dir:target. Например,

```
PATCH_DEPENDS=  ${NONEXISTENT}:${PORTSDIR}/java/jfc:extract
```

будет переходить в подкаталог java/jfc вашего дерева портов для распаковки.

Зависимость проверяется внутри цели `patch`. Часть *target* может быть опущена, если она совпадает с `DEPENDS_TARGET`.

5.8.7. USES

Могут быть добавлены параметры для определения различных характерных особенностей и зависимостей, которыми обладает данный порт. Они указываются путём добавления в Makefile этой строки:

```
USES= feature[:arguments]
```

Для получения полного списка значений смотрите USES.

Значение USES нельзя присваивать после подключения bsd.port.pre.mk.

5.8.8. USE_*

Для определения общих зависимостей, совместно используемых многими портами, предназначено несколько переменных. Их использование является необязательным, но помогает упростить избыточность файлов Makefile порта. Каждый из них оформляется как USE_*. Эти переменные можно использовать только в Makefile порта и ports/Mk/bsd.*.mk. Они не предназначены для установки пользователями параметров - используйте для этих целей PORT_OPTIONS.

Установка любых USE_* в /etc/make.conf *всегда* является ошибочным действием. В частности, установка

```
USE_GCC=X.Y
```

(где X.Y соответствует версии) добавит зависимость от gccXY к каждому порту, включая и сам lang/gccXY!

*Таблица 2. Переменные USE_**

Переменная	Значение
USE_BZIP2	tar-архивы порта упакованы при помощи bzip2.
USE_ZIP	tar-архивы порта упакованы при помощи zip.

Переменная	Значение
USE_GCC	Для сборки порта требуется GCC (gcc или g++). Некоторым портам подходит любая версия, для других требуются последние современные версии. Обычно используется со значением any (в этом случае используется встроенный GCC в тех версиях FreeBSD, в состав которых он всё ещё входит, или устанавливается порт lang/gcc, когда Clang является компилятором C/C++ по умолчанию) или yes (всегда используется стабильная современная версия GCC из порта lang/gcc). Также в значении переменной можно указать точную версию, например 4.7. Минимально допустимую версию можно указать как 4.6+. GCC из основной системы используется в случае, если его версия удовлетворяет запрошенной, иначе собирается подходящая версии компилятора из порта с соответствующей коррекцией переменных CC и CXX.

Переменные, относящиеся к gmake и сценарию configure, описаны в Механизмы построения, а autoconf, automake и libtool описаны в Использование GNU Autotools. Переменные, связанные с Perl, описаны в Использование Perl. Переменные X11 перечислены в Использование X11. Использование GNOME работает с переменными GNOME и Использование KDE с KDE. Использование Java описывает переменные Java, а Веб-приложения содержит информацию об Apache, PHP и модулях PEAR. Python обсуждается в Использование Python, а Ruby в Использование Ruby. Использование SDL предоставляет переменные, используемые для приложений SDL, и, наконец, Использование Xfce содержит информацию о приложении Xfce.

5.8.9. Минимальная версия зависимости

Минимальная версия зависимости может быть указана в любой переменной *_DEPENDS, за исключением LIB_DEPENDS, с использованием следующего синтаксиса:

```
p5-Spiffy>=0.26:${PORTSDIR}/devel/p5-Spiffy
```

Первое поле содержит название зависимого пакета, которое обязано совпадать с записью в базе данные пакетов, знак сравнения и версию пакета. Зависимость удовлетворяется, если на машине установлен p5-Spiffy-0.26 или новее.

5.8.10. Замечания касательно зависимостей

Как уже отмечено выше, целью, которая вызывается по умолчанию в случае, когда это

требует зависимость, является DEPENDS_TARGET. Она по умолчанию есть install. Это пользовательская переменная; она нигде не определена в файле Makefile порта. Если вашему порту требуется особый метод обработки зависимости, воспользуйтесь частью :target переменной *_DEPENDS вместо того, чтобы переопределять DEPENDS_TARGET.

Когда вы набираете команду make clean, эта операция также выполняется и над зависимостями этого порта. Если вы не хотите, чтобы это случилось, определите переменную NOCLEANDEPENDS в вашем окружении. Это может быть особенно нужным, если порт имеет нечто, что занимает много времени на построение, в своём списке зависимостей, например, KDE, GNOME или Mozilla.

Чтобы безусловно зависеть от другого порта, укажите переменную ${NONEXISTENT} в качестве первого поля переменной BUILD_DEPENDS или RUN_DEPENDS. Пользуйтесь этим, только когда вам нужно иметь исходный код другого порта. Вы можете сэкономить время на компиляции, указав также и цель. Например,

```
BUILD_DEPENDS=  ${NONEXISTENT}:${PORTSDIR}/graphics/jpeg:extract
```

всегда будет переходить в каталог с портом jpeg и распаковывать его.

5.8.11. Зацикленные зависимости фатальны

Не помещайте зацикливающиеся зависимости в дерево портов!

Технология построения портов не защищена от зацикленных зависимостей. Если вы создадите такую, то у кого-нибудь и где-нибудь установка FreeBSD будет немедленно сломана, а у остальных сломается несколько позже. Это на самом деле очень трудно распознать; если вы сомневаетесь, то перед внесением изменений проверьте, что выполнили следующее: cd /usr/ports; make index. Этот процесс может быть достаточно медленным на старых машинах, хотя вы сможете спасти большое количество людей-включая себя-от грядущих бед.

5.8.12. Автоматические зависимости и проблемы, которые они вызывают

Зависимости должны быть указаны либо явно, либо с использованием фреймворка OPTIONS. Использование прочих методов, таких как автоматическое обнаружение зависимостей, усложняет индексирование, что вызывает проблемы в управлении портами и пакетами.

Пример 8. Некорректное объявление необязательной зависимости

```
.include <bsd.port.pre.mk>

.if exists(${LOCALBASE}/bin/foo)
LIB_DEPENDS=    libbar.so:${PORTSDIR}/foo/bar
```

```
.endif
```

Проблема автоматического добавления зависимостей заключается в том, что файлы и настройки за пределами порта могут произвольно меняться. Пример: после построения индекса устанавливается набор портов. При этом один из них устанавливает проверяемый файл. На этом этапе индекс будет неправильным, потому что установленный порт неожиданно получит новую зависимость. Индекс может быть по прежнему неправильным даже после его перестроения, в случае если другие порты также определят дополнительные зависимости, основываясь на существовании других файлов.

Пример 9. Корректное объявление необязательной зависимости

```
OPTIONS_DEFINE= BAR
BAR_DESC=   Bar support

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MBAR}
LIB_DEPENDS=    libbar.so:${PORTSDIR}/foo/bar
.endif
```

Правильным способом является проверка переменных параметров. Этот способ не приводит к несоответствиям в индексе набора портов, поскольку параметры определены до построения индекса. При этом можно использовать простые скрипты для автоматизации построения, установки и обновления этих портов и соответствующих им пакетов.

5.8.13. USE_ и WANT_

Переменные USE_ задаются мейнтейнером порта для определения программного обеспечения, от которого этот порт зависит. Порт, для которого нужен Firefox, укажет

```
USE_FIREFOX=    yes
```

Некоторые переменные USE_ могут принимать номера версий или другие параметры. Например, порт, который требует Apache 2.2, укажет

```
USE_APACHE= 22
```

В некоторых случаях для большего контроля над зависимостями используются переменные WANT_, которые позволяют указывать требования в более точной форме. Например, взгляните на порт mail/squirrelmail. Этому порту нужны несколько модулей PHP, которые перечислены в переменной USE_PHP:

```
USE_PHP=    session mhash gettext mbstring pcre openssl xml
```

Эти модули доступны в версиях CLI и web, поэтому версия web выбрана с переменной `WANT_`:

```
WANT_PHP_WEB=   yes
```

Имеющиеся переменные `USE_` и `WANT_` определены в файлах в /usr/ports/Mk.

5.9. MASTERDIR

Если вашему порту требуется построение довольно различающихся версий пакетов через переменную (задающую, например, разрешение, или размер бумаги), которая принимает различные значения, создайте для каждого пакета отдельный подкаталог, чтобы пользователям было легче определить, каким пакетом воспользоваться, но попробуйте использовать совместно между портами как можно больше файлов. В типичном случае вам потребуются только очень короткие файлы Makefile во всех каталогах, кроме одного, если вы будете использовать переменные с умом. В отдельных файлах Makefile вы можете использовать переменную `MASTERDIR` для указания каталога, в котором находятся все остальные файлы. Также используйте переменную как часть `PKGNAMESUFFIX`, чтобы пакеты имели разные имена.

Продемонстрируем это на примере. Вот часть файла japanese/xdvi300/Makefile:

```
PORTNAME=   xdvi
PORTVERSION=    17
PKGNAMEPREFIX=  ja-
PKGNAMESUFFIX=  ${RESOLUTION}
 :
# default
RESOLUTION?=    300
.if ${RESOLUTION} != 118 && ${RESOLUTION} != 240 && \
    ${RESOLUTION} != 300 && ${RESOLUTION} != 400
    @${ECHO_MSG} "Error: invalid value for RESOLUTION: \"${RESOLUTION}\""
    @${ECHO_MSG} "Possible values are: 118, 240, 300 (default) and 400."
    @${FALSE}
.endif
```

Порт japanese/xdvi300 содержит также все обычные патчи, файлы для пакета и так далее. Если вы введете здесь команду `make`, она возьмет в качестве разрешения значение по умолчанию (300) и построит порт обычным образом.

Для другого разрешения приведем *полный*xdvi118/Makefile:

```
RESOLUTION= 118
MASTERDIR=  ${.CURDIR}/../xdvi300
```

```
.include "${MASTERDIR}/Makefile"
```

(xdvi240/Makefile и xdvi400/Makefile похожи). Задание MASTERDIR говорит bsd.port.mk, что обычный набор подкаталогов типа FILESDIR и SCRIPTDIR находится в каталоге xdvi300. Строчка RESOLUTION=118 переопределят строку RESOLUTION=300 в файле xdvi300/Makefile и порт будет построен с разрешением 118.

5.10. Страницы Справочника

Если ваш порт определяет корнем для файлов Справочника каталог, отличный от PREFIX, вы можете использовать переменную MANDIRS, чтобы указать эти каталоги. Обратите внимание, что файлы страниц справочника следует размещать в pkg-plist наряду с остальными файлами. MANDIRS предназначена для автоматического сжатия страниц справочника, так чтобы имена файлов оканчивались на .gz.

5.11. Файлы в формате info

Если в вашем пакете нужна установка файлов GNU info, они должны быть перечислены в переменной INFO (без окончания .info), по записи на документ. Предполагается, что эти файлы устанавливаются в PREFIX/INFO_PATH. Вы можете изменить INFO_PATH, если ваш пакет использует другое место для размещения. Однако, это не рекомендуется делать. Эти записи всего лишь содержат путь относительно PREFIX/INFO_PATH. Например, lang/gcc34 устанавливает файлы info в PREFIX/INFO_PATH/gcc34, и в INFO будет что-то вроде этого:

```
INFO=   gcc34/cpp gcc34/cppinternals gcc34/g77 ...
```

Перед регистрацией пакета соответствующий код установки/удаления будет автоматически добавлен во временный pkg-plist.

5.12. Опции для Makefile

Многие приложения могут быть построены в различных конфигурациях и с дополнительной функциональностью. Например, выбор естественного (человеческого) языка, GUI против командной строки или типа используемой базы данных. Пользователи могут нуждаться в различных конфигурациях, отличных от используемой по умолчанию, поэтому в системе портов предусмотрен механизм, позволяющий автору порта управлять сборкой того или иного варианта конфигурации. Правильная поддержка этих необязательных параметров облегчает пользователям жизнь и даёт два или более порта по цене одного.

5.12.1. Knobs

5.12.1.1. WITH* и WITHOUT*

Эти переменные предназначены для установки системным администратором. Многие из них стандартизованы в файле ports/KNOBS.

При создании порта не давайте имя для knob, специфичное для данного приложения. На примере порта Avahi, используйте `WITHOUT_MDNS` вместо `WITHOUT_AVAHI_MDNS`.

Не стоит рассчитывать, что `WITH* обязательно имеет соответствующую переменную WITHOUT*`, и наоборот. В общем случае, предполагается значение по умолчанию.

Если обратное не указано, то проверяется только факт установки самих переменных, но не их конкретное значение типа `YES` или `NO`.

Таблица 3. Основные переменные `WITH* и WITHOUT*`

Переменная	Значение
`WITH_OPENSSL_BASE`	Использовать версию OpenSSL из базовой системы.
`WITH_OPENSSL_PORT`	Устанавливает версию OpenSSL из security/openssl, даже если в базовой системе последняя версия.

5.12.1.2. Наименование KNOBS

Портеры должны использовать так называемые knobs для помощи конечным пользователям и для поддержания количества наименований knobs в небольшом количестве. Список популярных названий knobs можно найти в файле KNOBS

Названия knobs должны отражать, что это такое и что выполняет. Если у порта имеется библиотечный префикс в `PORTNAME`, то он должен присутствовать в названии knobs.

5.12.2. `OPTIONS`

5.12.2.1. Описание

При установке порта переменные `OPTIONS_*` предоставляют пользователю окно диалога с отображением доступных параметров, с записью выбранных параметров в файл /var/db/ports/${UNIQUENAME}/options. Эти опции повторно используются при следующем построении порта.

Когда пользователь запускает `make config` (или запускает впервые `make build`), инфраструктура выполняет проверку существования файла /var/db/ports/${UNIQUENAME}/options. Если этот файл не существует, то используются значения `OPTIONS_*` и отображается диалоговое окно, в котором эти параметры можно включить или выключить. Затем сохраняется файл опций options, и выбранные переменные используются при построении порта.

Если новая версия порта добавляет новые значения `OPTIONS`, то пользователю будет представлено окно диалога с сохраненными заполненными значениями старых `OPTIONS.`

`make showconfig` отображает сохраненную конфигурацию. Для удаления сохраненной

конфигурации используйте `make rmconfig`.

5.12.2.2. Синтаксис

`OPTIONS_DEFINE` содержит список используемых `OPTIONS`. Они независимы друг от друга и не сгруппированы:

```
OPTIONS_DEFINE= OPT1 OPT2
```

Далее после определения следует описание `OPTIONS` (не является обязательным, но настоятельно рекомендуется):

```
OPT1_DESC=  Describe OPT1
OPT2_DESC=  Describe OPT2
OPT3_DESC=  Describe OPT3
OPT4_DESC=  Describe OPT4
OPT5_DESC=  Describe OPT5
OPT6_DESC=  Describe OPT6
```

ports/Mk/bsd.options.desc.mk содержит описание множества наиболее используемых `OPTIONS`; переопределять их, как правило, не нужно.

При описании параметров старайтесь представить себя на месте пользователя: "Что это делает?" и "Для чего бы я захотел включить это?" Не делайте простое повторение названия. Например, описание параметра `NLS` как "include NLS support" ("включить поддержку NLS") не поможет пользователю, который уже видит название параметра, но может не знать, что это означает. Описав его как "Native Language Support via gettext utilities" ("Поддержка национального языка через утилиты gettext"), вы поможете пользователю гораздо больше.

`OPTIONS` можно группировать в виде переключателей, для которых разрешен выбор единственного варианта в каждой группе:

```
OPTIONS_SINGLE=     SG1
OPTIONS_SINGLE_SG1= OPT3 OPT4
```

`OPTIONS` можно группировать в виде переключателей, для которых разрешен выбор единственного варианта (или ни одного) в каждой группе:

```
OPTIONS_RADIO=      RG1
OPTIONS_RADIO_RG1=  OPT7 OPT8
```

`OPTIONS` также можно группировать в виде списков со множественным выбором, для

которых обязан быть включен *по крайней мере один* из параметров:

```
OPTIONS_MULTI=      MG1
OPTIONS_MULTI_MG1=  OPT5 OPT6
```

OPTIONS также можно группировать в виде списков со множественным выбором, для которых могут быть включены любые параметры, включая отсутствие выбора:

```
OPTIONS_GROUP=      GG1
OPTIONS_GROUP_GG1=  OPT9 OPT10
```

По умолчанию OPTIONS находится в выключенном положении, если при этом оно также отсутствует в списке OPTIONS_DEFAULT:

```
OPTIONS_DEFAULT=    OPT1 OPT3 OPT6
```

Определения OPTIONS обязаны быть до подключения bsd.port.options.mk. Переменные PORT_OPTIONS могут быть проверены только после подключения bsd.port.options.mk. Вместо этого также можно использовать подключение bsd.port.pre.mk, что все еще широко используется в портах, написанных до появления bsd.port.options.mk. Но имейте в виду, что некоторые переменные, обычно, это некоторые флаги USE_*, после подключения bsd.port.pre.mk будут работать не так, как этого от них ожидают.

Пример 10. Простое использование OPTIONS

```
OPTIONS_DEFINE= FOO BAR
FOO_DESC=   Enable option foo
BAR_DESC=   Support feature bar

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MFOO}
CONFIGURE_ARGS+=--with-foo
.else
CONFIGURE_ARGS+=--without-foo
.endif

.if ${PORT_OPTIONS:MBAR}
RUN_DEPENDS+=   bar:${PORTSDIR}/bar/bar
.endif

.include <bsd.port.mk>
```

Пример 11. Проверка незаданных значений `OPTIONS`

```
.if ! ${PORT_OPTIONS:MEXAMPLES}
CONFIGURE_ARGS+=--without-examples
.endif
```

Пример 12. Пример реального использования `OPTIONS`

```
OPTIONS_DEFINE=     EXAMPLES

OPTIONS_SINGLE=     BACKEND
OPTIONS_SINGLE_BACKEND= MYSQL PGSQL BDB

OPTIONS_MULTI=      AUTH
OPTIONS_MULTI_AUTH= LDAP PAM SSL

EXAMPLES_DESC=      Install extra examples
MYSQL_DESC=     Use MySQL as backend
PGSQL_DESC=     Use PostgreSQL as backend
BDB_DESC=       Use Berkeley DB as backend
LDAP_DESC=      Build with LDAP authentication support
PAM_DESC=       Build with PAM support
SSL_DESC=       Build with OpenSSL support

OPTIONS_DEFAULT=    PGSQL LDAP SSL

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MPGSQL}
USE_PGSQL=      yes
CONFIGURE_ARGS+=    --with-postgres
.else
CONFIGURE_ARGS+=    --without-postgres
.endif

.if ${PORT_OPTIONS:MICU}
LIB_DEPENDS+=   libicuuc.so:${PORTSDIR}/devel/icu
.endif

.if ! ${PORT_OPTIONS:MEXAMPLES}
CONFIGURE_ARGS+=    --without-examples
.endif

# Проверка других параметров OPTIONS

.include <bsd.port.mk>
```

5.12.2.3. Параметры по умолчанию

Следующие параметры по умолчанию всегда включены.

- `DOCS` - построение и установка документации.
- `NLS` - интернационализация.
- `EXAMPLES` - построение и установка примеров использования.
- `IPV6` - поддержка протокола IPv6.

Нет необходимости добавлять эти параметры в `OPTIONS_DEFAULT`. Тем не менее, чтобы отобразить их в окне диалога выбора параметров, они должны быть добавлены в `OPTIONS_DEFINE`.

5.12.3. Функция автоматической активации

При использовании сценария GNU configure, следите за тем, какие необязательные функции задействуются посредством автоматической активации. Отключайте явным образом те необязательные функции, которые вы не хотели бы использовать, через передачу соответствующих `--without-xxx` или `--disable-xxx` в переменной `CONFIGURE_ARGS`.

Пример 13. Неправильное управление опцией

```
.if ${PORT_OPTIONS:MFOO}
LIB_DEPENDS+=       libfoo.so:${PORTSDIR}/devel/foo
CONFIGURE_ARGS+=    --enable-foo
.endif
```

В приведенном выше примере представьте себе библиотеку libfoo, установленную в системе. Пользователь не желает, чтобы приложение использовало libfoo, и поэтому он выключает соответствующую опцию в диалоге `make config`. Но сценарий configure приложения определяет наличие библиотеки в системе и включает ее поддержку в итоговый исполняемый файл. Теперь, когда пользователь решит удалить libfoo из системы, система портов позволит это сделать (т.к. зависимость от libfoo не была записана), но приложение перестанет работать.

Пример 14. Правильное управление опцией

```
.if ${PORT_OPTIONS:MFOO}
LIB_DEPENDS+=       libfoo.so:${PORTSDIR}/devel/foo
CONFIGURE_ARGS+=    --enable-foo
.else
CONFIGURE_ARGS+=    --disable-foo
.endif
```

Во втором примере библиотека libfoo отключена явным образом. Сценарий configure не включает соответствующие функции в приложении, несмотря на присутствие библиотеки в системе.

При определенных условиях сокращенный синтаксис записи условий может вызывать проблемы со сложными конструкциями. Если вы получаете ошибки, такие как `Malformed conditional`, то может быть использован альтернативный синтаксис.

```
.if !empty(VARIABLE:MVALUE)
# as an alternative to
.if ${VARIABLE:MVALUE}
```

5.12.4. Вспомогательные макросы

Существует несколько макросов, упрощающих запись условных значений, которые отличаются в зависимости от набора параметров.

Если переменная `OPTIONS_SUB` имеет значение `yes`, то каждый из указанных в `OPTIONS_DEFINE` параметров будет добавлен в `PLIST_SUB`. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPTIONS_SUB=    yes
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
PLIST_SUB+= OPT1=""
.else
PLIST_SUB+= OPT1="@comment "
.endif
```

`X_CONFIGURE_ENABLE` дописывает в `CONFIGURE_ARGS` строку `--enable-${X_CONFIGURE_ENABLE}` или `--disable-${X_CONFIGURE_ENABLE}` в соответствии с состоянием `X`. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_ENABLE=  test
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --enable-test
.else
CONFIGURE_ARGS+=    --disable-test
.endif
```

X_CONFIGURE_WITH дописывает в CONFIGURE_ARGS строку --with-${X_CONFIGURE_WITH} или --without-${X_CONFIGURE_WITH} в соответствии с состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_WITH=    test
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --with-test
.else
CONFIGURE_ARGS+=    --without-test
.endif
```

Значение переменной X_CONFIGURE_ON будет дописано в CONFIGURE_ARGS в соответствии с состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_ON=  --add-test
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --add-test
.endif
```

Значение переменной X_CONFIGURE_OFF будет дописано в CONFIGURE_ARGS в соответствии с

состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CONFIGURE_OFF= --no-test
```

соответствует:

```
OPTIONS_DEFINE= OPT1
.include <bsd.port.options.mk>
.if ! ${PORT_OPTIONS:MOPT1}
CONFIGURE_ARGS+=    --no-test
.endif
```

Значение переменной X_CMAKE_ON будет дописано в CMAKE_ARGS в соответствии с состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CMAKE_ON=  -DTEST:BOOL=true
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
CMAKE_ARGS+=    -DTEST:BOOL=true
.endif
```

Значение переменной X_CMAKE_OFF будет дописано в CMAKE_ARGS в соответствии с состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_CMAKE_OFF= -DTEST:BOOL=false
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ! ${PORT_OPTIONS:MOPT1}
CMAKE_ARGS+=    -DTEST:BOOL=false
.endif
```

Для любой из следующих переменных:

- ALL_TARGET
- CATEGORIES
- CFLAGS
- CPPFLAGS
- CXXFLAGS
- CONFIGURE_ENV
- DISTFILES
- EXTRA_PATCHES
- INSTALL_TARGET
- LDFLAGS
- MAKE_ARGS
- MAKE_ENV
- PATCH_SITES
- PATCHFILES
- PLIST_FILES
- PLIST_DIRS
- PLIST_DIRSTRY
- USES

Значение переменной X_ABOVEVARIABLE будет дописано в ABOVEVARIABLE в соответствии с состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_USES=  gmake
OPT1_CFLAGS=    -DTEST
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
USES+=      gmake
CFLAGS+=    -DTEST
.endif
```

Если установлена X_ABOVEVARIABLE_OFF, то флаг ABOVEVARIABLE будет автоматически выставлен

при выключенном параметре X. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_USES_OFF=gmake
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ! ${PORT_OPTIONS:MOPT1}
USES+=  gmake
.endif
```

Для любого из следующих типов зависимости:

- PKG_DEPENDS
- EXTRACT_DEPENDS
- PATCH_DEPENDS
- FETCH_DEPENDS
- BUILD_DEPENDS
- LIB_DEPENDS
- RUN_DEPENDS

Значение переменной X_ABOVEVARIABLE будет дописано в ABOVEVARIABLE в соответствии с состоянием X. Следующая запись:

```
OPTIONS_DEFINE= OPT1
OPT1_LIB_DEPENDS=   liba.so:${PORTSDIR}/devel/a
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MOPT1}
LIB_DEPENDS+=   liba.so:${PORTSDIR}/devel/a
.endif
```

Если установлена X_ABOVEVARIABLE_OFF, то зависимость типа ABOVEVARIABLE будет добавлена при выключенном параметре X. Например:

```
OPTIONS_DEFINE= OPT1
OPT1_LIB_DEPENDS_OFF= liba.so:${PORTSDIR}/devel/a
```

соответствует:

```
OPTIONS_DEFINE= OPT1

.include <bsd.port.options.mk>

. if ! ${PORT_OPTIONS:MOPT1}
LIB_DEPENDS+=   liba.so:${PORTSDIR}/devel/a
.endif
```

5.13. Задание рабочего каталога

Каждый порт распаковывается в рабочий каталог, который должен быть доступным для записи. В системе портов по умолчанию `DISTFILES` распаковываются в каталог с именем `${DISTNAME}`. Другими словами, если вы задали:

```
PORTNAME=   foo
PORTVERSION=    1.0
```

то дистрибутивные файлы порта содержат каталог верхнего уровня, foo-1.0, и все файлы расположены в этом каталоге.

Если это не ваш случай, то имеется несколько переменных, которые вы можете переопределить.

5.13.1. `WRKSRC`

Эта переменная задаёт имя каталога, который создаётся при распаковке исходных файлов приложения. В нашем предыдущем примере если бы распаковка происходила в каталог с именем foo (а не foo-1.0), то вы должны написать:

```
WRKSRC= ${WRKDIR}/foo
```

или, как вариант

```
WRKSRC= ${WRKDIR}/${PORTNAME}
```

5.13.2. `NO_WRKSUBDIR`

Если порт вообще не распаковывается ни в какой каталог, то вы должны задать для этого

переменную NO_WRKSUBDIR.

```
NO_WRKSUBDIR=   yes
```

5.14. Разрешение конфликтов

Для регистрации конфликта между пакетами и портами используются три различные переменные: CONFLICTS, CONFLICTS_INSTALL и CONFLICTS_BUILD.

Переменные регистрации конфликта автоматически определяют переменную IGNORE, которая более подробно описана в Пометка неустанавливаемого порта как BROKEN, FORBIDDEN или IGNORE .

При удалении одного из конфликтующих портов целесообразно сохранить записи CONFLICTS в тех других портах в течении нескольких месяцев, чтобы позаботиться о тех пользователей, которые обновляются от случая к случаю.

5.14.1. CONFLICTS_INSTALL

Если ваш пакет не может существовать вместе с другими (из-за конфликта файлов, несовместимости времени выполнения и так далее), перечислите имена остальных пакетов в переменной CONFLICTS_INSTALL. Здесь вы можете использовать шаблоны командного интерпретатора, такие как * и ?. Имена пакетов должны выглядеть так же, как в /var/db/pkg. Пожалуйста, убедитесь, что CONFLICTS_INSTALL не содержит пакет самого этого порта. В противном случае не будет работать установка с использованием переменной FORCE_PKG_REGISTER. Проверка CONFLICTS_INSTALL выполняется после процесса сборки и до процесса установки.

5.14.2. CONFLICTS_BUILD

Если ваш порт не может быть собран, когда уже установлен другой, перечислите имена остальных портов в переменной CONFLICTS_BUILD. Здесь вы можете использовать шаблоны командного интерпретатора, такие как * и ?. Имена пакетов должны выглядеть так же, как в /var/db/pkg. Проверка CONFLICTS_BUILD выполняется до процесса сборки. Конфликты сборки в получаемом пакете не записываются.

5.14.3. CONFLICTS

Если ваш порт не может быть собран, когда уже установлен другой, а получаемый пакет не может существовать вместе с другими, перечислите имена остальных пакетов в переменной CONFLICTS. Здесь вы можете использовать шаблоны командного интерпретатора, такие как * и ?. Имена пакетов должны выглядеть так же, как в /var/db/pkg. Пожалуйста, убедитесь, что CONFLICTS не содержит пакет самого этого порта. В противном случае не будет работать установка с использованием переменной FORCE_PKG_REGISTER. Проверка CONFLICTS выполняется до процессов сборки и установки.

5.15. Установка файлов

5.15.1. Макросы INSTALL_*

Используйте макросы, которые есть в файле bsd.port.mk для обеспечения правильных прав доступа файлов в целях `*-install` порта. Устанавливайте права владения напрямую в pkg-plist через соответствующие записи `@owner owner` и `@group group`. Эти операторы работают до момента их переопределения или до конца pkg-plist, поэтому не забывайте их сбрасывать, когда они больше не нужны. По умолчанию владение устанавливается для `root:wheel`.

- `INSTALL_PROGRAM` - это команда для установки бинарных выполнимых файлов.
- `INSTALL_SCRIPT` - это команда для установки выполнимых скриптов.
- `INSTALL_LIB` - это команда для установки динамических библиотек.
- `INSTALL_KLD` - это команда для установки загружаемых модулей ядра. Некоторые архитектуры предпочитают, чтобы для модулей сохранялись отладочные сведения, по этой причине используйте эту команду вместо `INSTALL_PROGRAM`.
- `INSTALL_DATA` - это команда для установки совместно используемых файлов данных.
- `INSTALL_MAN` - это команда для установки страниц Справочника и другой документации (никаких файлов она не сжимает).

В основе работы этих макросов лежит команда `install` со всеми соответствующими флагами. Смотрите пример их использования ниже.

5.15.2. Удаление отладочной информации в бинарных файлах и динамических библиотеках

Не удаляйте отладочную информацию из бинарных файлов вручную, если вы это делали. Во всех двоичных файлах отладочная информация должна быть удалена, и макрос `INSTALL_PROGRAM` выполнит установку и удаление отладочной информации одновременно (обратитесь к следующему разделу). Макрос `INSTALL_LIB` делает то же самое для динамических библиотек.

Если вам нужно удалить отладочную информацию из файла без использования макросов `INSTALL_PROGRAM` и `INSTALL_LIB`, то это можно сделать при помощи `${STRIP_CMD}`. Обычно это делается внутри цели `post-install`. К примеру:

```
post-install:
    ${STRIP_CMD} ${STAGEDIR}${PREFIX}/bin/xdl
```

Удаление отладочной информации из нескольких файлов:

```
post-install:
    .for l in geometry media body track world
    ${STRIP_CMD} ${STAGEDIR}${PREFIX}/lib/lib${PORTNAME}-${l}.so.0
```

```
    .endfor
```

Для проверки того, удалена ли отладочная информация из файла, используйте file(1). Для двоичных файлов file(1) печатает `stripped` или `not stripped`. Кроме того, strip(1) определяет, была ли уже удалена из программы отладочная информация, и в этом случае просто завершает свою работу.

5.15.3. Установка целого дерева файлов

Иногда должно быть установлено большое количество файлов с сохранением их иерархической организации. Например, копирование дерева каталогов целиком из `WRKSRC` в целевой каталог внутри `PREFIX`. Обратите внимание, что `PREFIX`, `EXAMPLESDIR`, `DATADIR` и другие переменные пути всегда должны предваряться `STAGEDIR`, чтобы не ломать staging (смотрите Staging).

Для этой ситуации существует два макроса. Преимущество от использования этих макросов вместо команды `cp` в том, что они гарантируют установку правильного владельца и прав на конечные файлы. Первый макрос, `COPYTREE_BIN`, делает все устанавливаемые файлы исполняемыми, что подходит для установки в PREFIX/bin. Второй макрос, `COPYTREE_SHARE`, не устанавливает на файлы права исполнения, и, таким образом, подходит для установки файлов внутри каталога PREFIX/share.

```
post-install:
    ${MKDIR} ${STAGEDIR}${EXAMPLESDIR}
    (cd ${WRKSRC}/examples && ${COPYTREE_SHARE} . ${STAGEDIR}${EXAMPLESDIR})
```

В этом примере устанавливается содержимое каталога examples из установочных файлов производителя в надлежащее место для примеров вашего порта.

```
post-install:
    ${MKDIR} ${STAGEDIR}${DATADIR}/summer
    (cd ${WRKSRC}/temperatures && ${COPYTREE_SHARE} "June July August"
${STAGEDIR}${DATADIR}/summer)
```

А в этом примере будут установлены данные летних месяцев в подкаталог summer каталога DATADIR.

В качестве третьего параметра в макросе `COPYTREE_*` можно передать дополнительные параметры `find`. Например, чтобы в первом примере установить все файлы кроме файлов Makefile, можно использовать следующую команду.

```
post-install:
    ${MKDIR} ${STAGEDIR}${EXAMPLESDIR}
    (cd ${WRKSRC}/examples && \
    ${COPYTREE_SHARE} . ${STAGEDIR}${EXAMPLESDIR} "! -name Makefile")
```

Эти макросы не производят добавление устанавливаемых файлов в pkg-plist. Они должны быть добавлены туда вручную. Необязательные файлы документации (PORTDOCS, смотрите Установка дополнительной документации) и примеров (PORTEXAMPLES) всегда должны предваряться в pkg-plist префиксами %%PORTDOCS%% или %%PORTEXAMPLES%%.

5.15.4. Установка дополнительной документации

Если с вашим программным обеспечением поставляется некоторая документация, отличающаяся от стандартных страниц Справочника и файлов info, которая, как вы думаете, будет полезна пользователям, установите ее в каталог PREFIX/shared/doc. Это может быть сделано, как и в предыдущем разделе, в цели post-install.

Создайте для вашего порта новый каталог. Имя каталога должно соответствовать тому, что представляет из себя порт. Обычно это означает PORTNAME. Однако, если вы думаете, что пользователь захочет иметь разные версии порта, установленные одновременно, то вы можете использовать полное имя PKGNAME.

Поскольку устанавливаются только файлы, перечисленные в pkg-plist, безопасным способом будет устанавливать документацию в STAGEDIR всегда (смотрите Staging). Следовательно, блоки .if нужны только для файлов достаточно большого размера, установка которых влечёт значительные накладные расходы на операции ввода/вывода.

```
post-install:
    ${MKDIR} ${STAGEDIR}${DOCSDIR}
    ${INSTALL_MAN} ${WRKSRC}/docs/xvdocs.ps ${STAGEDIR}${DOCSDIR}
```

Вот несколько полезных переменных и то, как они преобразуются по умолчанию при использовании в Makefile:

- DATADIR преобразуется в PREFIX/shared/PORTNAME.
- DATADIR_REL преобразуется в share/PORTNAME.
- DOCSDIR преобразуется в PREFIX/shared/doc/PORTNAME.
- DOCSDIR_REL преобразуется в share/doc/PORTNAME.
- EXAMPLESDIR преобразуется в PREFIX/shared/examples/PORTNAME.
- EXAMPLESDIR_REL преобразуется в share/examples/PORTNAME.

> Параметр DOCS управляет установкой дополнительной документации в DOCSDIR. Это не относится к стандартным страницам справочника и страницам info. Все, что устанавливается в DATADIR и EXAMPLESDIR, соответственно управляется через параметры DATA и EXAMPLES.

Эти переменные экспортируются в PLIST_SUB. Их значения появятся там в виде имён путей относительно PREFIX, если это возможно. То есть share/doc/PORTNAME в списке сборки по умолчанию будет заменен на %%DOCSDIR%%, и так далее. (Дополнительную информацию о подстановке в pkg-plist можно найти здесь.)

Все условно устанавливаемые файлы и каталоги с документацией должны быть перечислены в файле pkg-plist с префиксом %%PORTDOCS%%, например:

```
%%PORTDOCS%%%%DOCSDIR%%/AUTHORS
%%PORTDOCS%%%%DOCSDIR%%/CONTACT
%%PORTDOCS%%@dirrm %%DOCSDIR%%
```

В качестве альтернативы перечислению файлов документации в файле pkg-plist, порт может указать в переменной PORTDOCS список имён файлов и глобальных шаблонов командного процессора для добавления в окончательный список сборки. Имена будут задаваться относительно DOCSDIR. Таким образом, порт, использующий PORTDOCS и нестандартное местоположение документации, должен задавать соответствующим образом и DOCSDIR. Если каталог указан в PORTDOCS или соответствует шаблону для этой переменной, то полное поддерево с входящими в него файлами и каталогами будет регистрироваться в окончательном списке сборки. Если параметр DOCS не задан, то файлы и каталоги, перечисленные в PORTDOCS, не будут установлены и добавлены в список сборки порта. Установка документации в PORTDOCS, как это показано выше, остаётся за самим портом. Типичный пример использования PORTDOCS выглядит следующим образом:

```
PORTDOCS=   README.* ChangeLog docs/*
```

Эквивалентами PORTDOCS для файлов, устанавливаемых в DATADIR и EXAMPLESDIR являются PORTDATA и PORTEXAMPLES соответственно.

Во время установки выводится содержимое pkg-message. За подробной информацией обратитесь к разделу об использовании pkg-message. Файл pkg-message не нужно добавлять в pkg-plist.

5.15.5. Подкаталоги внутри PREFIX

Попробуйте поместить все файлы порта в правильных подкаталогах каталога PREFIX. Некоторые порты игнорируют все установки и помещают все в подкаталог с именем порта, что неправильно. Также многие порты помещают все, кроме бинарных файлов, файлов заголовков и страниц Справочника, в подкаталог каталога lib, что не очень хорошо работает с подходом BSD. Многие файлы должны быть перемещены в одно из следующих местоположений: etc (настроечные/конфигурационные файлы), libexec (выполнимые файлы, запускаемые из других программ), sbin (исполнимые файлы для администраторов/менеджеров системы), info (документация в формате info для просмотрщика info) или share (независимые от архитектуры файлы). Обратитесь к hier(7) для прояснения деталей; правила, покрывающие /usr, достаточно хорошо подходят также и к /usr/local. Исключением являются порты, имеющие дело с "новостями" USENET. Они могут использовать каталог PREFIX/news для установки своих файлов.

Глава 6. Особые соглашения

Имеется ещё несколько вещей, которые вы должны иметь в виду при создании порта. Этот раздел описывает наиболее часто встречающиеся из них.

6.1. Staging

bsd.port.mk ожидает от портов работу с "каталогом сборки". Это означает, что порт должен устанавливать файлы не напрямую в назначенные каталоги (то есть, например, под PREFIX), а в отдельный каталог, из которого затем собирается пакет. Во многих случаях привилегии root для этого не требуются, что делает возможным сборку пакетов из-под непривилегированного пользователя. В режиме staging порт собирается и устанавливается в каталог сборки STAGEDIR. Пакет создается из каталога сборки и затем устанавливается в систему. В инструментарии automake такая концепция именуется DESTDIR; в прочем, в FreeBSD DESTDIR имеет собственное значение (смотрите PREFIX и DESTDIR).

Если для порта всё ещё требуются системные привилегии при выполнении цели package, то в Makefile должна быть добавлена следующая строка:

```
NEED_ROOT=  yes
```

Метапорты, то есть порты, которые не устанавливают файлы непосредственно, а только зависят от других портов, должны по возможности избегать распаковки mtree(8) в каталог сборки. Это основная иерархия каталогов пакета, и эти пустые каталоги будут выглядеть лишними. Для предотвращения распаковки mtree(8) добавьте эту строку:

```
NO_MTREE=   yes
```

Staging задействуется посредством добавления переменной STAGEDIR слева от путей, которые используются в целях pre-install, do-install и post-install (смотрите примеры в книге). Обычно сюда относятся PREFIX, ETCDIR, DATADIR, EXAMPLESDIR, MANPREFIX, DOCSDIR и так далее. Каталоги должны создаваться при выполнении цели post-install. Избегайте использования абсолютных путей, когда это возможно.

При создании символический ссылки STAGEDIR должен ставиться только для пути назначения. Например:

```
${LN} -sf libfoo.so.42 ${STAGEDIR}${PREFIX}/lib/libfoo.so
```

Первоначальный путь ${PREFIX}/lib/libfoo.so.42 выглядит нормально, но по факту может быть неправильным. Абсолютные пути могут указывать на неподходящее место, например, когда удалённая файловая система смонтирована по NFS как непривилегированная точка монтирования. Относительные пути реже подвержены проблемам и часто намного короче.

Порты, устанавливающие модули ядра, должны предварять путь установки (по умолчанию

/boot/modules) переменной STAGEDIR.

6.2. Динамические библиотеки

Если ваш порт устанавливает одну или несколько динамических библиотек, определите переменную USE_LDCONFIG, которая приведёт к запуску из bsd.port.mk команды ${LDCONFIG} -m относительно каталога, в который устанавливается новая библиотека (как правило, это PREFIX/lib), во время выполнения цели post-install для её регистрации в кэше динамических библиотек. Эта переменная, если она определена, также приведёт к добавлению соответствующей пары команд @exec /sbin/ldconfig -m и @unexec /sbin/ldconfig -R в ваш файл pkg-plist, так что пользователь, устанавливающий пакет, сможет сразу же использовать динамическую библиотеку, а удаление пакета не приведёт к тому, что система будет предполагать, что библиотека всё ещё имеется в наличии.

```
USE_LDCONFIG=   yes
```

Если нужно, вы можете переопределить каталог по умолчанию, задав значение USE_LDCONFIG, в котором должны быть перечислены каталоги, в которые устанавливаются динамические библиотеки. Например, если ваш порт устанавливает динамические библиотеки в каталоги PREFIX/lib/foo и PREFIX/lib/bar, то вы можете в файле Makefile указать следующее:

```
USE_LDCONFIG=   ${PREFIX}/lib/foo ${PREFIX}/lib/bar
```

Будьте добры перепроверить, т.к. часто это вовсе не является необходимым и может быть решено иначе с помощью -rpath или установки LD_RUN_PATH во время компоновки (для примера смотрите lang/moscow_ml), или с помощью сценария-обёртки, который выставляет LD_LIBRARY_PATH перед запуском исполняемого файла как это делает www/seamonkey.

При установке 32-разрядных библиотек на 64-разрядной системе используйте вместо этого USE_LDCONFIG32.

Постарайтесь сохранять номера версий динамических библиотек в формате libfoo.so.0. Наш компоновщик позаботится только о старшем (первом) номере.

Если при обновлении порта увеличивается старший номер версии библиотеки, то для всех портов, компонуемых с затронутой библиотекой, следует увеличить значение PORTREVISION для форсирования перекомпиляции с новой версией библиотеки.

6.3. Порты с ограничениями на распространение или с правовым обременением

Лицензии бывают разных видов, и некоторые накладывают ограничение на то, как приложение может быть оформлено в виде пакета, может ли оно продаваться для извлечения коммерческой выгоды, и так далее.

На вас, как на человека, портирующего приложение, ложится обязанность прочесть лицензионные соглашения на программное обеспечение и удостовериться, что проект FreeBSD не будет являться их нарушителем, если будет заниматься распространением исходного кода или в бинарном виде по FTP/HTTP или на CD-ROM. Если у вас возникли сомнения, то, пожалуйста, обратитесь в Список рассылки, посвящённый Портам FreeBSD.

В подобных ситуациях можно использовать переменные, описываемые в последующих разделах.

6.3.1. NO_PACKAGE

Эта переменная указывает, что мы не можем создавать для приложения двоичный пакет. К примеру, лицензия не позволяет бинарное распространение или она может запрещать распространение пакетов, созданных из изменённых исходников.

Однако файлы DISTFILES могут свободно зеркалироваться по FTP/HTTP. Они также могут распространяться, используя CD-ROM (или на похожих носителях), если не установлена переменная NO_CDROM.

NO_PACKAGE должна также использоваться, если двоичный пакет, как правило, бесполезен, а приложение должно всегда компилироваться из исходного кода. К примеру, если в приложение во время компиляции жёстко включается конфигурационная информация, привязанная к конкретной системе, то задайте переменную NO_PACKAGE.

Значением переменной NO_PACKAGE должна быть строка, описывающая причину, по которой пакет не должен создаваться.

6.3.2. NO_CDROM

Эта переменная указывает на то, что, хотя мы имеем право создавать бинарные пакеты, мы не можем помещать эти пакеты или файлы DISTFILES порта на CD-ROM (или на похожие носители) для перепродажи. Однако бинарные пакеты и файлы DISTFILES порта будут оставаться доступными посредством FTP/HTTP.

Если эта переменная устанавливается вместе с NO_PACKAGE, то только файлы порта DISTFILES будут доступны, и только посредством FTP/HTTP.

В качестве значения NO_CDROM должна указываться строка, описывающая причины, по которым порт не может распространяться на CD-ROM. К примеру, это применяется, если лицензионное соглашение приложения предполагает только его "некоммерческое" использование.

6.3.3. NOFETCHFILES

Файлы, определенные в переменной NOFETCHFILES, не будут извлекаться ни из одного из MASTER_SITES. Примером такого файла является файл, поставляемый на CD-ROM.

Инструменты, проверяющие доступность этих файлов на MASTER_SITES, должны

игнорировать эти файлы и не сообщать о них.

6.3.4. RESTRICTED

Задайте эту переменную, если лицензия на приложение не позволяет ни зеркалировать файлы DISTFILES, ни распространять бинарный пакет через FTP/HTTP или на CD-ROM.

Ни NO_CDROM, ни NO_PACKAGE не стоит устанавливать вместе с RESTRICTED, так как последняя переменная подразумевает первые две.

В качестве значения RESTRICTED должна указываться строка, описывающая причины, по которым порт нельзя распространять. Обычно это означает, что порт использует закрытое программное обеспечение, а пользователь должен вручную сгрузить файлы DISTFILES, возможно, после заполнения регистрационной формы или подтверждения соглашения с условиями EULA.

6.3.5. RESTRICTED_FILES

Если заданы RESTRICTED или NO_CDROM, то значение этой переменной по умолчанию соответствует ${DISTFILES} ${PATCHFILES}, в противном случае она пуста. Если ограничены в распространении лишь некоторые из дистрибутивных файлов, то в этой переменной задаётся их список.

6.3.6. LEGAL_TEXT

Если порт имеет правовое обременение, которое не покрывается перечисленными выше переменными, то переменной LEGAL_TEXT следует присвоить строку с описанием данного обременения. Например, если было получено особое разрешение для FreeBSD на распространение двоичного файла, то эта переменная должна содержать соответствующее указание.

6.3.7. /usr/ports/LEGAL и LEGAL

Порт, содержащий любую из перечисленных выше переменных, также должен быть добавлен в /usr/ports/LEGAL. Первый столбец содержит шаблон совпадения с дистрибутивными файлами, имеющими ограничения на распространение. Второй столбец содержит корень порта. Третий столбец содержит вывод make -VLEGAL.

6.3.8. Примеры использования

Предпочтительным способом реализации утверждения "архивы исходных текстов для этого порта должны загружаться самостоятельно" является следующее:

```
.if !exists(${DISTDIR}/${DISTNAME}${EXTRACT_SUFX})
IGNORE= may not be redistributed because of licensing reasons. Please visit some-
website to accept their license and download ${DISTFILES} into ${DISTDIR}
.endif
```

Это одновременно и информирует пользователя, и устанавливает нужные метаданные на пользовательской машине для использования автоматическими программами.

Обратите внимание, что данная кляуза должна предшествовать подключению файла bsd.port.pre.mk.

6.4. Механизмы построения

6.4.1. Параллельное построение портов

Инфраструктура портов FreeBSD поддерживает параллельное построение с использованием множественных подпроцессов `make`, что позволяет системам SMP задействовать всю доступную мощность CPU, тем самым делая построение портов более быстрым и эффективным.

Это достигается путём передачи флага `-jX` команде make(1). Такое построение портов является поведением по умолчанию. К сожалению, не все порты поддерживают параллельную сборку достаточно хорошо, и поэтому может потребоваться выключить этот механизм явным образом путём добавления переменной `MAKE_JOBS_UNSAFE=yes`. Эта переменная используется в случае, когда известно, что порт ломается с `-jX`.

6.4.2. `make`, `gmake` и `imake`

Если ваш порт использует GNU make, то установите `USES= gmake`.

Таблица 4. Переменные для портов, использующих gmake

Переменная	Значение
`USES= gmake`	Для сборки порта требуется `gmake`.
`GMAKE`	Полный путь к команде `gmake`, если отсутствует в `PATH`.

Если ваш порт является приложением X, которое создает файлы Makefile из Imakefile, используя imake, то установите `USES= imake`. Это заставит стадию конфигурирования автоматически выполнить `xmkmf -a`. Если флаг `-a` представляет для вашего порта проблему, то установите `XMKMF=xmkmf`. Если порт использует imake, но не понимает цель `install.man`, то следует установить `NO_INSTALL_MANPAGES=yes`.

Если исходный Makefile вашего порта имеет что-нибудь помимо `all` в качестве основной цели построения, то задайте соответствующее значение `ALL_TARGET`. То же касается `install` и `INSTALL_TARGET`.

6.4.3. Сценарий `configure`

Если ваш порт использует сценарий `configure` для получения файлов Makefile из файлов Makefile.in, то установите `GNU_CONFIGURE=yes`. Если вы хотите дать дополнительные параметры сценарию `configure` (аргументом по умолчанию является `--prefix=${PREFIX} --infodir=${PREFIX}/${INFO_PATH} --mandir=${MANPREFIX}/man --build=${CONFIGURE_TARGET}`),

установите эти параметры в CONFIGURE_ARGS. Дополнительные переменные окружения можно передать, используя переменную CONFIGURE_ENV.

Таблица 5. Переменные для портов, использующих configure

Переменная	Значение
GNU_CONFIGURE	Порт использует сценарий configure для подготовки построения.
HAS_CONFIGURE	То же, что и GNU_CONFIGURE, кроме того, что цель configure по умолчанию не добавляется в CONFIGURE_ARGS.
CONFIGURE_ARGS	Дополнительные параметры, передаваемые сценарию configure.
CONFIGURE_ENV	Дополнительные переменные окружения, задаваемые для запуска сценария configure.
CONFIGURE_TARGET	Переопределить цель configure по умолчанию. Значением по умолчанию является ${MACHINE_ARCH}-portbld-freebsd${OSREL}.

6.4.4. Использование cmake

Если порт использует CMake, определите USES= cmake или USES= cmake:outsource для построения во внешнем каталоге (см. ниже).

Таблица 6. Переменные для портов, использующих cmake

Переменная	Значение
CMAKE_ARGS	Специфичные для порта флаги CMake, передаваемые cmake.
CMAKE_BUILD_TYPE	Тип построения (предопределённые профили построения CMake). По умолчанию Release, Debug при использовании WITH_DEBUG.
CMAKE_ENV	Переменные окружения для передачи cmake. По умолчанию ${CONFIGURE_ENV}.
CMAKE_SOURCE_PATH	Путь к каталогу с исходным кодом. По умолчанию ${WRKSRC}.

Таблица 7. Переменные построения cmake, устанавливаемые пользователем

Переменная	Значение
CMAKE_VERBOSE	Разрешает подробный вывод сообщений при построении. Значение по умолчанию не задано, если не заданы BATCH или PACKAGE_BUILDING.

Переменная	Значение
CMAKE_NOCOLOR	Запрещает цветной вывод сообщений при построении. Значение по умолчанию не задано, если не заданы BATCH или PACKAGE_BUILDING.

CMake поддерживает следующие профили построения: Debug, Release, RelWithDebInfo и MinSizeRel. Профили Debug и Release учитывают системные флаги *FLAGS; RelWithDebInfo и MinSizeRel соответственно определяют CFLAGS со значением -O2 -g и -Os -DNDEBUG. Значение CMAKE_BUILD_TYPE экспортируется в нижнем регистре в PLIST_SUB и должно использоваться, если порт устанавливает файлы *.cmake в зависимости от типа построения (для примера посмотрите на deskutils/strigi). Следует учитывать, что некоторые проекты могут определять собственные профили построения и/или форсировать конкретный тип построения через установку CMAKE_BUILD_TYPE в файлах CMakeLists.txt. Для того чтобы порт для такого проекта учитывал CFLAGS и WITH_DEBUG, из этих файлов должны быть удалены значения CMAKE_BUILD_TYPE.

Большинство проектов, основанных на CMake, поддерживают метод внешнего (out-of-source) построения. Для порта внешнее построение можно запросить с использованием суффикса :outsource. В этом случае CONFIGURE_WRKSRC, BUILD_WRKSRC и INSTALL_WRKSRC будут иметь значение ${WRKDIR}/.build для каталога, содержащего файлы, получаемые на этапах конфигурации и построения; при этом каталог с исходным кодом будет оставаться без изменений.

Пример 15. Пример для USES= cmake

Следующий отрывок демонстрирует использование CMake для порта. CMAKE_SOURCE_PATH обычно не требуется, но может быть установлен, когда исходный код не находится в верхнем каталоге или если порт используется для построения части проекта.

```
USES=           cmake:outsource
CMAKE_SOURCE_PATH=  ${WRKSRC}/subproject
```

6.4.5. Использование scons

Если ваш порт использует SCons, определите USE_SCONS=yes.

Таблица 8. Переменные для портов, использующих scons

Переменная	Значение
SCONS_ARGS	Специфичные для порта флаги SCons, передаваемые окружению SCons.
SCONS_BUILDENV	Переменные для установки в системном окружении.

Переменная	Значение
SCONS_ENV	Переменные для установки в окружении SCons.
SCONS_TARGET	Последний параметр для передачи SCons, похожий на MAKE_TARGET.

Для того, чтобы сторонний SConstruct соответствовал всему, что передается SCons в переменной SCONS_ENV (самое главное, это CC/CXX/CFLAGS/CXXFLAGS), примените патч к SConstruct, так чтобы переменная построения Environment выглядела следующим образом:

```
env = Environment(**ARGUMENTS)
```

В дальнейшем ее можно изменить при помощи env.Append и env.Replace.

6.5. Использование GNU Autotools

6.5.1. Введение

Различные инструменты GNU autotools предоставляют механизм абстракции для построения частей программного обеспечения на широком наборе операционных систем и аппаратных архитектур. Внутри Коллекции Портов отдельный порт может использовать эти инструменты при помощи простых конструкций:

```
USE_AUTOTOOLS=  tool:version[:operation] ...
```

К моменту написания *tool* может быть одним из libtool, libltdl, autoconf, autoheader, automake или aclocal.

version указывает конкретную версию используемого инструмента (смотрите devel/{automake,autoconf,libtool}[0-9]+ для получения действительных версий).

operation является необязательным расширением и указывает на способ использования инструмента.

Одновременно может быть указано несколько инструментов, добавляя их все на одной строке или используя конструкцию Makefile +=.

В заключение, существует специальный инструмент по называнию autotools, который является удобной функцией при установке всех доступных версий autotools для возможности проведения кросс-разработки. Это также может быть достигнуто путем установки порта devel/autotools.

6.5.2. libtool

Динамические библиотеки, использующие инфраструктуру построения GNU, обычно используют libtool для настройки компиляции и установки динамических библиотек в

соответствии с особенностями данной операционной системы. В типичной практике используется копирование встроенного в приложение libtool. Если вам нужно использовать внешнюю команду libtool, то вы можете использовать версию, поставляемую Коллекцией Портов:

```
USE_AUTOTOOLS=  libtool:version[:env]
```

При отсутствии дополнительных операций, libtool:version сообщает инфраструктуре построения о применении патча к сценарию configure с установленной в системе копией libtool. Означает использование GNU_CONFIGURE. Более того, некоторые переменные make и оболочки shell будут назначены для дальнейшего использования этим портом. Подробности смотрите в bsd.autotools.mk.

При использовании операции :env будет настроено только окружение.

Наконец, LIBTOOLFLAGS и LIBTOOLFILES можно установить по желанию, чтобы переопределить наиболее вероятные аргументы для libtool и файлы, предназначенные для изменения. Большинству портов это скорее всего не понадобится. Для дальнейших подробностей смотрите bsd.autotools.mk.

6.5.3. libltdl

Некоторые порты задействуют пакет с библиотекой libltdl, которая является частью комплекта libtool. Использование этой библиотеки не вызывает автоматическое использование самой libtool, и, таким образом, обеспечивается отдельная конструкция.

```
USE_AUTOTOOLS=  libltdl:version
```

Всё, что в настоящее время она делает, это добавление LIB_DEPENDS для подходящего порта libltdl, потому она предоставляется как удобная функция для помощи в устранении всяких зависимостей от портов autotools вне инфраструктуры USE_AUTOTOOLS. Для этого инструмента не существует необязательных операций.

6.5.4. autoconf и autoheader

Вместо сценария configure некоторые порты содержат шаблон autoconf в файле configure.ac. Вы можете использовать следующие присвоения, чтобы позволить autoconf создать сценарий configure, а autoheader создать заголовки шаблона для использования в сценарии configure.

```
USE_AUTOTOOLS=  autoconf:version[:env]
```

и

```
USE_AUTOTOOLS=  autoheader:version
```

которые также подразумевают использование `autoconf:version`.

Аналогично команде `libtool` включение необязательной операции `:env` всего лишь настраивает окружение для дальнейшего использования. Без этого выполняется наложение патчей и переконфигурирование порта.

Дополнительные необязательные переменные `AUTOCONF_ARGS` и `AUTOHEADER_ARGS` можно переопределить в Makefile порта, если указано явным образом. Как и с эквивалентами `libtool`, большинству портов это вряд ли понадобится.

6.5.5. `automake` и `aclocal`

Некоторые пакеты содержат только файлы Makefile.am. Они должны быть преобразованы в файлы Makefile.in с использованием automake и дальнейшей обработкой `configure` для получения настоящего Makefile.

Аналогично, иногда пакеты не поставляются с вложенными файлами aclocal.m4, снова требуемых для построения программного обеспечения. Их можно получить командой `aclocal`, которая просматривает configure.ac или configure.in.

`aclocal` имеет похожую связь с `automake`, как у `autoheader` с `autoconf`, что описано в предыдущей главе. `aclocal` подразумевает использование `automake`, таким образом, мы имеем:

```
USE_AUTOTOOLS=  automake:version[:env]
```

и

```
USE_AUTOTOOLS=  aclocal:version
```

которые также подразумевают использование `automake:version`.

Также как и для `libtool` и `autoconf`, подключение необязательной операции `:env` всего лишь устанавливает окружение для дальнейшего пользования. Без этого выполняется реконфигурирование этого порта.

Как и в случае с `autoconf` и `autoheader`, обе команды `automake` и `aclocal` соответственно имеют необязательные переменные `AUTOMAKE_ARGS` и `ACLOCAL_ARGS`, которые при необходимости можно переопределить в Makefile порта.

6.6. Использование GNU `gettext`

6.6.1. Простой вариант использования

Если для вашего порта требуется `gettext`, добавьте `USES= gettext`, и ваш порт унаследует зависимость от devel/gettext. `USES` содержит перечень других значений для использования `gettext`.

Довольно распространенным случаем является использование в порте `gettext` и `configure`. Как правило, GNU `configure` способен находить `gettext` автоматически. Если он все же не сможет это сделать, то подсказки для размещения `gettext` можно передать через переменные окружения `CPPFLAGS` и `LDFLAGS`:

```
USES=   gettext
CPPFLAGS+=  -I${LOCALBASE}/include
LDFLAGS+=   -L${LOCALBASE}/lib

GNU_CONFIGURE=  yes
```

Конечно же, этот код можно записать в более компактном виде, если передавать флаги в `configure` не требуется:

```
USES=   gettext
GNU_CONFIGURE=  yes
```

6.6.2. Оптимальное использование

Некоторые программные продукты позволяют отключать NLS, к примеру, передавая параметр `--disable-nls` сценарию `configure`. В этом случае ваш порт должен использовать `gettext`, в зависимости от значения `NLS`. Для портов небольшой или средней сложности вы можете полагаться на следующую идиому:

```
GNU_CONFIGURE=  yes

.include <bsd.port.options.mk>

.if ${PORT_OPTIONS:MNLS}
USES+=          gettext
PLIST_SUB+=     NLS=""
.else
CONFIGURE_ARGS+=    --disable-nls
PLIST_SUB+=     NLS="@comment "
.endif

.include <bsd.port.mk>
```

Следующий пункт в вашем списке дел разобраться, чтобы файлы каталога сообщения включались в список упаковки по условию. Часть, входящая в Makefile, уже обеспечена этой идиомой. Остальное объясняется в главе продвинутые практики pkg-plist. Вкратце, каждое вхождение `%%NLS%%` в pkg-plist будет заменено на “@comment”, если NLS выключен, или пустой строкой, если включен. В результате строки, предваряемые `%%NLS%%`, станут комментариями в итоговом листе упаковки, если NLS выключен; иначе, префикс будет просто удален. Всё, что вам нужно, это вставить `%%NLS%%` перед каждым путем к файлу каталога сообщений в pkg-plist. Например:

```
%%NLS%%share/locale/fr/LC_MESSAGES/foobar.mo
%%NLS%%share/locale/no/LC_MESSAGES/foobar.mo
```

В особо сложных случаях вам понадобиться использовать более продвинутые техники, чем данный рецепт, такие как динамические списки упаковки.

6.6.3. Управление каталогами сообщений

Существует момент, который следует учитывать при установке файлов каталогов сообщений. Целевые каталоги для размещения, расположенные под LOCALBASE/shared/locale, редко когда должны создаваться и удаляться портом. Для наиболее популярных языков имеются собственные каталоги, перечисленные в PORTSDIR/Templates/BSD.local.dist. Каталоги для множества других языков управляются с помощью порта devel/gettext. Обратите внимание на его pkg-plist и посмотрите, куда данный порт собирается установить файлы каталогов сообщений для единственного в своем роде языка.

6.7. Использование Perl

Если `MASTER_SITES` установлена в значение `MASTER_SITE_PERL_CPAN`, то предпочтительным значением `MASTER_SITE_SUBDIR` является имя иерархии верхнего уровня. Например, рекомендуемым значением для `p5-Module-Name` является `Module`. Иерархию верхнего уровня можно посмотреть на сайте cpan.org. Это поддерживает порт в рабочем состоянии при изменении модуля автором.

Исключением этого правила является отсутствие соответствующего каталога или файла с дистрибутивом в этом каталоге. В качестве `MASTER_SITE_SUBDIR` в этом случае разрешается использовать id автора.

В качестве значения все из настраиваемых knobs ниже принимают `YES` или строку с версией вида `5.8.0+`. `YES` означает, что данный порт можно использовать с любой из поддерживаемых версий Perl. Если порт работает только с некоторыми версиями Perl, то это можно обозначить при помощи строки с версией, указывающей на минимальную версию (пример: `5.7.3+`), максимальную версию (пример: `5.8.0-`) или точную версию (пример: `5.8.3`).

Таблица 9. Переменные для портов, использующих Perl

Переменная	Значение
`USE_PERL5`	Perl 5 используется для построения и работы.
`USE_PERL5_BUILD`	Perl 5 используется для построения.
`USE_PERL5_RUN`	Perl 5 используется во время работы.
`PERL`	Полный путь к интерпретатору Perl 5, либо в системе, либо установленному из портов, но без номера версии. Используйте это, если вам нужно заменить строки “#!” в скриптах.

Переменная	Значение
PERL_CONFIGURE	Конфигурация при помощи MakeMaker языка Perl. Влечёт USE_PERL5.
PERL_MODBUILD	Конфигурация, построение и установка с использованием Module::Build. Влечёт PERL_CONFIGURE.

Порты для модулей Perl, которые не имеют официального вебсайта, должны указывать cpan.org в строке WWW в файле pkg-descr. Предпочтительная форма URL http://search.cpan.org/dist/Module-Name/ (включая завершающий слэш).

Не используйте ${SITE_PERL} в объявлении зависимостей. Использование этой конструкции подразумевает наличие подключенного bsd.perl.mk, что не всегда так. Порты, зависимые от этого порта, получат неправильные зависимости, если файлы этого порта будут перемещены при последующем обновлении. Правильный способ объявления зависимостей для модулей Perl показан в примере ниже.

Пример 16. Пример зависимости Perl

```
p5-IO-Tee>=0.64:${PORTSDIR}/devel/p5-IO-Tee
```

Для портов Perl, которые устанавливают страницы справочника, в pkg-plist можно использовать макрос PERL5_MANx (где *x* принимает значение от 1 до 9). Например,

```
lib/perl5/5.14/man/man3/AnyEvent::I3.3.gz
```

можно заменить на

```
%%PERL5_MAN3%%/AnyEvent::I3.3.gz
```

6.8. Использование X11

6.8.1. Компоненты X.Org

X.Org является реализацией X11, доступной в Коллекции Портов. Если ваше приложение зависит от компонентов X, установите в переменную USE_XORG в перечень требуемых компонентов. К настоящему времени доступными компонентами являются:

bigreqsproto compositeproto damageproto dmx dmxproto dri2proto evieproto fixesproto fontcacheproto fontenc fontsproto fontutil glproto ice inputproto kbproto libfs oldx pciaccess pixman printproto randrproto recordproto renderproto resourceproto scrnsaverproto sm trapproto

videoproto x11 xau xaw xaw6 xaw7 xbitmaps xcmiscproto xcomposite xcursor xdamage xdmcp xevie xext xextproto xf86bigfontproto xf86dgaproto xf86driproto xf86miscproto xf86rushproto xf86vidmodeproto xfixes xfont xfontcache xft xi xinerama xineramaproto xkbfile xkbui xmu xmuu xorg-server xp xpm xprintapputil xprintutil xproto xproxymngproto xrandr xrender xres xscrnsaver xt xtrans xtrap xtst xv xvmc xxf86dga xxf86misc xxf86vm.

Всегда актуальный перечень можно найти в /usr/ports/Mk/bsd.xorg.mk.

Проект Mesa является попыткой обеспечить свободную реализацию OpenGL. Вы можете указать зависимость от различных компонентов этого проекта при помощи переменной USE_GL. Действительные опции: glut, glu, glw, glew, gl и linux. Для обратной совместимости значение yes соответствует glu.

Пример 17. Пример для USE_XORG

```
USE_XORG=   xrender xft xkbfile xt xaw
USE_GL=     glu
```

Таблица 10. Переменные для портов, использующих X

USES= imake	Используется imake.
XMKMF	Задаёт маршрут до xmkmf, если он отсутствует в PATH. По умолчанию это xmkmf -a.

Пример 18. Использование переменных X11 в порте

```
# Использовать некоторые библиотеки X11
USE_XORG=   x11 xpm
```

6.8.2. Порты, которым требуется Motif

Если вашему порту требуется Motif, задайте переменную USES= motif в файле Makefile. Реализация Motif, используемая по умолчанию, находится в x11-toolkits/open-motif. Пользователи вместо этого могут выбрать x11-toolkits/lesstif через установку переменной WANT_LESSTIF.

Переменная MOTIFLIB будет установлена в bsd.port.mk, чтобы ссылаться на соответствующую библиотеку Motif. Пожалуйста, измените исходные тексты вашего порта на использование ${MOTIFLIB} везде, где упоминается библиотека Motif, в первоначальном Makefile или Imakefile.

Существует два общих случая:

- Если порт обращается к библиотеке Motif как -lXm в своих файлах Makefile или Imakefile, просто подставьте вместо этих обращений ${MOTIFLIB}.
- Если порт использует XmClientLibs в своем файле Imakefile, измените это обращение на

${MOTIFLIB} ${XTOOLLIB} ${XLIB}.

Заметьте, что переменная MOTIFLIB (как правило) раскрывается в -L/usr/local/lib -lXm или /usr/local/lib/libXm.a, так что нет нужды впереди добавлять -L или -l.

6.8.3. Шрифты для X11

Если ваш порт устанавливает шрифты для X Window System, поместите их в каталог LOCALBASE/lib/X11/fonts/local.

6.8.4. Получение поддельного DISPLAY, используя Xvfb

Некоторые приложения для успешной компиляции требуют наличие работающего дисплея X11. Это создает проблему для машин, которые работают в режиме headless. При использовании следующего канонического хака инфраструктура построения запустит сервер X в виртуальном фреймбуфере. Затем переменная работающего DISPLAY передается при построении.

```
USES=   display
```

6.8.5. Элементы рабочего стола

Элементы рабочего стола (стандарта Freedesktop) предоставляют способ автоматической настройки функций рабочего стола при установке новой программы, не требуя вмешательства пользователя. Например, новые программы автоматически отображаются в меню приложений совместимых окружений рабочего стола. Элементы рабочего стола изначально появились в окружении рабочего стола GNOME, но в настоящее время являются стандартом и также работают с KDE и Xfce. Такая небольшая автоматизация предоставляет реальное удобство для пользователя, и посему элементы рабочего стола приветствуются в приложениях, которые можно использовать в окружении рабочего стола.

6.8.5.1. Использование предопределенных файлов .desktop

Порты, включающие предопределенные файлы *.desktop, должны включать эти файлы в pkg-plist и устанавливать их в каталог $LOCALBASE/shared/applications. Для установки этих файлов используется макрос INSTALL_DATA.

6.8.5.2. Обновление базы данных рабочего стола

Если в файле порта portname.desktop имеется запись MimeType, то база данных рабочего стола олжна быть обновлена после установки и удаления. Для этого укажите USES= desktop-file-utils.

6.8.5.3. Создание элементов рабочего стола с использованием DESKTOP_ENTRIES

Элементы рабочего стола можно легко создавать для приложений, используя переменную DESKTOP_ENTRIES. Будет автоматически создан, установлен и добавлен в pkg-plist файл с названием name.desktop. Синтаксис:

```
DESKTOP_ENTRIES=    "NAME" "COMMENT" "ICON" "COMMAND" "CATEGORY" StartupNotify
```

Перечень возможных категорий доступен на вебсайте Freedesktop. `StartupNotify` отобразит, поддерживает ли приложение *уведомления о запуске*. Как правило, это графический индикатор часы вместо указателя мыши, меню или панель, которые уведомляют пользователя о загрузке программы. Программа, поддерживающая уведомления о запуске, очистит этот индикатор после запуска. Программы, несовместимые с уведомлениями о запуске, не будут очищать индикатор (возможно, вызывая путаницу и приводя пользователей в бешенство), и поэтому должны иметь `StartupNotify` в выключенном состоянии `false`; тогда индикатор не будет отображаться совсем.

Пример:

```
DESKTOP_ENTRIES=    "ToME" "Roguelike game based on JRR Tolkien's work" \
            "${DATADIR}/xtra/graf/tome-128.png" \
            "tome -v -g" "Application;Game;RolePlaying;" \
            false
```

6.9. Использование GNOME

Для задания того, какие компоненты GNOME использует конкретный порт, проект FreeBSD/GNOME использует собственный набор переменных. На странице проекта FreeBSD/GNOME размещён исчерпывающий список этих переменных.

6.10. Использование Qt

6.10.1. Порты, для которых требуется Qt

Таблица 11. Переменные для портов, использующих Qt

`USE_QT4`	Указывает инструменты и библиотеки в качестве зависимостей для портов, которые используют Qt 4. Для получения подробностей смотрите выбор компонентов Qt 4.
`QT_PREFIX`	Устанавливается в значение, содержащее путь к установленному Qt (переменная только для чтения).
`MOC`	Устанавливается в значение, содержащее путь к `moc` (переменная только для чтения). По умолчанию устанавливается в соответствии со значением `USE_QT_VER`.

QTCPPFLAGS	Дополнительные флаги компилятора для инструментального пакета Qt, передаваемые через переменную CONFIGURE_ENV. По умолчанию устанавливается в соответствии со значением USE_QT_VER.
QTCFGLIBS	Дополнительные флаги компоновки для инструментального пакета Qt, передаваемые через переменную CONFIGURE_ENV. По умолчанию устанавливается в соответствии со значением USE_QT_VER.
QTNONSTANDARD	Подавляет изменение CONFIGURE_ENV, CONFIGURE_ARGS, CPPFLAGS и MAKE_ENV.

Таблица 12. Дополнительные переменные для портов, использующих Qt 4.x

UIC	Устанавливает путь к uic (переменная только для чтения).
QMAKE	Устанавливает путь к qmake (переменная только для чтения).
QMAKESPEC	Устанавливает путь к конфигурационному файлу для qmake (переменная только для чтения).
QMAKEFLAGS	Дополнительные флаги для qmake.
QT_INCDIR	Устанавливает каталоги для заголовков Qt 4 (переменная только для чтения).
QT_LIBDIR	Устанавливает путь к библиотекам Qt 4 (переменная только для чтения).
QT_PLUGINDIRC	Устанавливает путь к плагинам Qt 4 (переменная только для чтения).

При заданной переменной USE_QT4 применяются следующие настройки:

```
CONFIGURE_ARGS+=    --with-qt-includes=${QT_INCDIR} \
            --with-qt-libraries=${QT_LIBDIR} \
            --with-extra-libs=${LOCALBASE}/lib \
            --with-extra-includes=${LOCALBASE}/include
CONFIGURE_ENV+= MOC="${MOC}" UIC="${UIC}" LIBS="${QTCFGLIBS}" \
        QMAKE="${QMAKE}" QMAKESPEC="${QMAKESPEC}" QTDIR="${QT_PREFIX}"
MAKE_ENV+=  QMAKESPEC="${QMAKESPEC}"

PLIST_SUB+= QT_INCDIR_REL=${QT_INCDIR_REL} \
        QT_LIBDIR_REL=${QT_LIBDIR_REL} \
        QT_PLUGINDIR_REL=${QT_PLUGINDIR_REL}
```

6.10.2. Выбор компонентов

В переменной USE_QT4 должны указываться зависимости от отдельных инструментов и библиотек Qt 4. К каждому компоненту можно добавить суффикс, _build или _run, отражающий, когда должна быть применена зависимость, во время сборки или выполнения, соответственно. Если суффикс отсутствует, зависимость от компонента будет и для времени сборки, и для времени выполнения. Обычно, компоненты библиотек должны указываться без суффиксов, компоненты инструментов - с суффиксом _build, а компоненты плагинов - с суффиксом _run. Наиболее общие используемые компоненты перечислены ниже (все доступные компоненты перечислены в _USE_QT4_ALL в файле /usr/ports/Mk/bsd.qt.mk):

Таблица 13. Доступные библиотечные компоненты Qt 4

Название	Описание
corelib	основная библиотека (можно опустить, если порт не использует ничего, кроме corelib)
gui	библиотека графического пользовательского интерфейса
network	сетевая библиотека
opengl	библиотека OpenGL
qt3support	библиотека совместимости с Qt 3
qtestlib	библиотека модульного тестирования
script	библиотека сценариев
sql	библиотека SQL
xml	библиотека XML

Вы можете определить, от каких библиотек зависит приложение, запустив ldd на основной исполняемый файл после успешной компиляции.

Таблица 14. Доступные компоненты инструментов Qt 4

Название	Описание
moc	мета-объектный компилятор (нужен при построении почти для каждого приложения Qt)
qmake	генератор Makefile / утилита построения
rcc	компилятор ресурсов (нужен, если приложение идет вместе с файлами **.rc** или.qrc)

Название	Описание
`uic`	компилятор пользовательского интерфейса (нужен, если приложение идет вместе с файлами *.ui, созданными при помощи Qt Designer, - на практике каждое приложение Qt с GUI)

Таблица 15. Доступные компоненты плагинов Qt 4

Название	Описание
`iconengines`	плагин для движка иконок SVG (если приложение поставляется с иконками SVG)
`imageformats`	плагины для графических форматов GIF, JPEG, MNG и SVG (если приложение поставляется с графическими файлами)

Пример 19. Выбор компонентов Qt 4

В этом примере портированное приложение использует библиотеку графического пользовательского интерфейса Qt 4, основную библиотеку Qt 4, все инструменты генерации кода Qt 4 и генератор Makefile Qt 4. Поскольку библиотека `gui` подразумевает зависимость от основной библиотеки, указывать `corelib` нет необходимости. Инструменты генерации кода Qt 4 `moc`, `uic` и `rcc`, а также генератор Makefile `qmake` нужны только для времени построения, поэтому они указаны с суффиксом `_build`:

```
USE_QT4=    gui moc_build qmake_build rcc_build uic_build
```

6.10.3. Использование `qmake`

Таблица 16. Переменные для портов, использующих `qmake`

Название	Описание
`QMAKE_ARGS`	Специцичные для порта флаги QMake для передачи программе `qmake`.
`QMAKE_ENV`	Переменные окружения, устанавливаемые для программы `qmake`. По умолчанию соответствует значению `${CONFIGURE_ENV}`.
`QMAKE_PRO`	Название файла проекта .pro. По умолчанию имеет пустое значение (с использованием автоопределения).

Если вместе с приложением вместо configure поставляется файл .pro, вы можете использовать следующее:

```
USES=   qmake
USE_QT4=    qmake_build
```

USES=qmake указывает порту на использование qmake в процессе конфигурации. Обратите внимание, что USES=qmake не подразумевает зависимость от Qt 4 qmake. Для этого в значении USE_QT4 должен присутствовать компонент qmake_build.

Приложения Qt часто пишутся в кроссплатформенной манере, и X11/Unix часто не является для них платформой разработки, что в свою очередь часто приводит к соответствующим упущенным моментам:

- *Отсутствующие дополнительные пути для заголовочных файлов.* Многие приложения идут с поддержкой иконки в системном трее, но пренебрегают смотреть на наличие заголовочных файлов и/или библиотеками в каталогах X11. Вы можете сообщить qmake, чтобы она добавила каталоги в пути поиска заголовочных файлов и библиотек через командную строку. К примеру:

  ```
  QMAKE_ARGS+= INCLUDEPATH+=${LOCALBASE}/include \
      LIBS+=-L${LOCALBASE}/lib
  ```

- *Фиктивные пути установки.* Иногда данные, такие как иконки и файлы .desktop, устанавливаются по умолчанию в каталоги, которые не просматриваются XDG-совместимыми приложениями. Примером является editors/texmaker - взгляните на patch-texmaker.pro из каталога files этого порта, который можно взять в качестве шаблона исправления этого непосредственно в файле проекта qmake.

6.11. Использование KDE

6.11.1. Задание переменных KDE 4

Если ваше приложение зависит от KDE 4.x, присвойте USE_KDE4 список требуемых компонентов. Для переопределения типа зависимости компонента могут быть использованы суффиксы _build и _run (например, baseapps_run). Если суффикс не задан, будет использован тип зависимости по умолчанию. Если вы хотите использовать оба типа, добавьте компонент дважды с обоими суффиксами (например, automoc4_build automoc4_run). Основные наиболее используемые компоненты перечислены ниже (актуальные компоненты задокументированы в начале файла /usr/ports/Mk/bsd.kde4.mk):

Таблица 17. Доступные компоненты KDE 4

Название	Описание
kdehier	Иерархия основных каталогов KDE
kdelibs	KDE Developer Platform
kdeprefix	Если установлено, то порт будет установлен в ${KDE4_PREFIX} вместо ${LOCALBASE}

Название	Описание
sharedmime	База данных MIME типов для портов KDE
automoc4	automoc для пакетов Qt 4
akonadi	Сервер хранения KDE-Pim
soprano	Фреймворк Qt 4 RDF
strigi	Поисковые даемон рабочего стола
libkcddb	Библиотека KDE CDDB
libkcompactdisc	Библиотека KDE для взаимодействия с аудио-CD
libkdeedu	Библиотеки, используемые для образовательных приложений
libkdcraw	Библиотека KDE LibRaw
libkexiv2	Библиотека KDE Exiv2
libkipi	KDE Image Plugin Interface
libkonq	Основная библиотека Konqueror
libksane	Библиотека KDE SANE ("Scanner Access Now Easy")
pimlibs	Библиотеки KDE-Pim
kate	Тектовый редактор
marble	Виртуальный глобус
okular	Универсальный просмотрщик документов
korundum	Привязка Ruby к KDE
perlkde	Привязка Perl к KDE
pykde4	Привязка Python к KDE
pykdeuic4	Компилятор пользовательского интерфейса PyKDE
smokekde	Библиотеки KDE SMOKE

Порты KDE 4.x устанавливаются в KDE4_PREFIX, что в настоящее время соответствует /usr/local/kde4. Это достигается путем указания компонента kdeprefix, который определяет значение по умолчанию для PREFIX. Тем не менее, порты учитывают любые PREFIX, установленные через переменную окружения MAKEFLAGS и/или параметры make.

Пример 20. Пример USE_KDE4

Это простой пример для порта KDE 4. USES= cmake:outsource указывает порту использовать CMake, конфигурационный инструмент, широко применяемый в проектах KDE 4 (подробное описание даёт Использование cmake). USE_KDE4 добавляет зависимость от библиотек KDE и заставляет порты использовать automoc4 во время

сборки. Требуемые компоненты KDE и другие зависимости можно определить в журнале configure. USE_KDE4 не подразумевает USE_QT4. Если порт требует какой-либо из компонентов Qt 4, их следует указать в USE_QT4.

```
USES=       cmake:outsource
USE_KDE4=   kdelibs kdeprefix automoc4
USE_QT4=    moc_build qmake_build rcc_build uic_build
```

6.12. Использование Java

6.12.1. Задание переменных

Если вашему порту необходимо наличие Java™ Development Kit (JDK™) для построения, работы или даже распаковки дистрибутивного файла, то в нём должна быть задана переменная USE_JAVA.

В Коллекции Портов присутствуют несколько JDK различных разработчиков и разных версий. Если ваш порт должен использовать одну из этих версий, то вы должны указать, какую именно. Самой последней версией и версией по умолчанию является java/openjdk6.

Таблица 18. Переменные, которые которые могут задаваться портами, использующими Java

Переменная	Значение
USE_JAVA	Должна быть определена для того, что последующие переменные вступили в действие.
JAVA_VERSION	Список версий Java, перечисленных через пробел, подходящих для порта. Опциональный знак `""` позволяет вам указать диапазон версий (возможные значения: `1.5[] 1.6[] 1.7[]`).
JAVA_OS	Список операционных систем, перечисленных через пробел, порты JDK для которых подходят для порта (возможные значения: native linux).
JAVA_VENDOR	Список разработчиков портов JDK, перечисленных через пробел, которые подходят для порта (возможные значения: freebsd bsdjava sun openjdk).
JAVA_BUILD	Если задана, то означает, что выбранный порт JDK должен быть добавлен к зависимостям порта для его построения.

Переменная	Значение
JAVA_RUN	Если задана, то означает, что выбранный порт JDK должен быть добавлен в зависимостям порта для его работы.
JAVA_EXTRACT	Если задана, то означает, что выбранный порт JDK должен быть добавлен в зависимостям порта для распаковки его дистрибутивных файлов.

Ниже перечисляются все значения, которые принимают переменные после задания переменной USE_JAVA:

Таблица 19. Переменные, доступные в портах, использующих Java

Переменная	Значение
JAVA_PORT	Название порта JDK (к примеру, 'java/openjdk6').
JAVA_PORT_VERSION	Полное наименовании версии порта JDK (к примеру, '1.6.0'). Если вам нужны только первые две цифры номера версии, используйте ${JAVA_PORT_VERSION:C/^([0-9])\.([0-9])(.*)$/\1.\2/}.
JAVA_PORT_OS	Операционная система, используемая портом JDK (к примеру, 'native').
JAVA_PORT_VENDOR	Разработчик порта JDK (к примеру, 'openjdk').
JAVA_PORT_OS_DESCRIPTION	Описание операционной системы, используемой портом JDK (к примеру, 'Native').
JAVA_PORT_VENDOR_DESCRIPTION	Описание разработчика порта JDK (к примеру, 'OpenJDK BSD Porting Team').
JAVA_HOME	Маршрут к установочному каталогу JDK (к примеру, '/usr/local/openjdk6').
JAVAC	Маршрут к используемому компилятору Java (к примеру, '/usr/local/openjdk6/bin/javac'.
JAR	Маршрут к используемой утилите jar (к примеру, '/usr/local/openjdk6/bin/jar' или '/usr/local/bin/fastjar').
APPLETVIEWER	Маршрут к утилите appletviewer (к примеру, '/usr/local/openjdk6/bin/appletviewer').
JAVA	Маршрут к выполняемому файлу java. Используйте его для запуска Java-программ (к примеру, '/usr/local/openjdk6/bin/java').

Переменная	Значение
JAVADOC	Маршрут к вспомогательной программе javadoc.
JAVAH	Маршрут к программе javah.
JAVAP	Маршрут к программе javap.
JAVA_KEYTOOL	Маршрут к вспомогательной программе keytool.
JAVA_N2A	Маршрут к утилите native2ascii.
JAVA_POLICYTOOL	Маршрут к программе policytool.
JAVA_SERIALVER	Маршрут к вспомогательной программе serialver.
RMIC	Маршрут к генератору каркаса программ RMI, утилите rmic.
RMIREGISTRY	Маршрут к программе регистрации RMI, rmiregistry.
RMID	Маршрут к программе-даемону RMI rmid.
JAVA_CLASSES	Маршрут к архиву, который содержит файлы классов JDK, ${JAVA_HOME}/jre/lib/rt.jar.

Вы можете воспользоваться make-целью java-debug для получения информации, необходимой для отладки вашего порта. При её выполнении будут выданы значения многих упомянутых выше переменных.

Кроме того, для единообразия установки всех портов Java определены следующие константы:

Таблица 20. Константы, определённые для портов, использующих Java

Константа	Значение
JAVASHAREDIR	Корневой каталог для всего, что связано с Java. По умолчанию: ${PREFIX}/shared/java.
JAVAJARDIR	Каталог, в который должны устанавливаться JAR-файлы. По умолчанию: ${JAVASHAREDIR}/classes.
JAVALIBDIR	Каталог, в который устанавливаются JAR-файлы из других портов. По умолчанию: ${LOCALBASE}/shared/java/classes.

Соответствующие записи определяются в обоих переменных PLIST_SUB (описана в Изменение содержимого pkg-plist в зависимости от make-переменных) и SUB_LIST.

6.12.2. Построение с Ant

Если построение порта производится с использованием Apache Ant, то необходимо определить `USE_ANT`. Таким образом Ant становится подкомандой make. Если в порте не определена цель `do-build`, то будет установлена цель по умолчанию, которая просто запускает Ant в соответствии со значением `MAKE_ENV`, `MAKE_ARGS` и `ALL_TARGET`. Это похоже на механизм `USES= gmake`, который описан в Механизмы построения.

6.12.3. Практические рекомендации

При портировании Java-библиотеки ваш порт должен устанавливать JAR-файл(ы) в каталог ${JAVAJARDIR}, а все остальные данные в каталог ${JAVASHAREDIR}/${PORTNAME} (за исключением документации, о которой пойдёт речь ниже). Для уменьшения размера упакованного файла вы можете сослаться на JAR-файл(ы) непосредственно в файле Makefile. Просто воспользуйтесь следующей директивой (в которой myport.jar является именем JAR-файла, устанавливаемого как часть порта):

```
PLIST_FILES+=   %%JAVAJARDIR%%/myport.jar
```

При портировании Java-приложения порт обычно устанавливает всё в один каталог (в том числе все свои JAR-зависимости). В этом отношении настоятельно рекомендуется использование ${JAVASHAREDIR}/${PORTNAME}. На усмотрение создателя порта остаётся решение вопроса о том, устанавливать ли дополнительные JAR-зависимости в этот каталог или напрямую использовать уже установленные (из каталога ${JAVAJARDIR}).

При портировании приложения Java™, для запуска сервиса которого требуется сервер приложений, такой как www/tomcat7, для производителя в порядке вещей является распространение файла .war. Файл .war - это Веб-приложение АРхивированное и оно распаковывается при вызове данным приложением. Избегайте добавлять файлы .war в pkg-plist. Это не является наилучшим решением. Сервер приложений производит расширение архива war без должной его очистки при удалении порта. Более подходящим способом работы с этим файлом будет распаковать архив, установить файлы и добавить их в pkg-plist.

```
TOMCATDIR=  ${LOCALBASE}/apache-tomcat-7.0
WEBAPPDIR=  myapplication

post-extract:
    @${MKDIR} ${WRKDIR}/${PORTDIRNAME}
    @${TAR} xf ${WRKDIR}/myapplication.war -C ${WRKDIR}/${PORTDIRNAME}

do-install:
    cd ${WRKDIR} && \
    ${INSTALL} -d -o ${WWWOWN} -g ${WWWGRP} ${TOMCATDIR}/webapps/${PORTDIRNAME}
    @cd ${WRKDIR}/${PORTDIRNAME} && ${COPYTREE_SHARE} \* ${WEBAPPDIR}/${PORTDIRNAME}
```

Вне зависимости от типа вашего порта (библиотека это или приложение), дополнительная документация должна устанавливаться в тоже самое место, что и для других портов.

Известно, что в зависимости от используемой версии JDK утилита JavaDoc генерирует различные наборы файлов. Для портов, которые не привязаны к использованию определённой версии JDK, таким образом становится проблематичным определить список файлов для упаковки (pkg-plist). Это одна из причин, по которой создателям портов настоятельно рекомендуется использовать макрос `PORTDOCS`. Более того, даже если вы сможете угадать набор файлов, который будет сгенерирован утилитой `javadoc`, размер получающегося файла pkg-plist голосует за использование `PORTDOCS`.

Значением по умолчанию для переменной `DATADIR` является ${PREFIX}/shared/${PORTNAME}. Хорошей идеей является переопределение для Java-портов значения `DATADIR` как ${JAVASHAREDIR}/${PORTNAME}. На самом деле `DATADIR` автоматически добавляется к `PLIST_SUB` (это описано в Изменение содержимого pkg-plist в зависимости от make-переменных), так что вы сможете использовать `%%DATADIR%%` непосредственно в pkg-plist.

Что касается выбора между построением портов Java из исходных текстов или их прямой установкой из бинарных дистрибутивов, то на момент создания этого текста определённой политики на этот счёт не существует. Однако участники Проекта FreeBSD Java рекомендуют создателям портов строить их из исходных текстов, если эта задача является несложной.

Все возможности, которые были описаны в этом разделе, реализованы в файле bsd.java.mk. Если вы предположите, что вашему порту требуется менее тривиальная поддержка Java, пожалуйста, взгляните сначала на журнал изменений bsd.java.mk в Subversion, так как для документирования последних изменений требуется какое-то время. Затем, если вы думаете, что не хватающая вам поддержка окажется полезной для многих других портов Java, обсудите ваш вопрос в freebsd-java.

Хотя в базе сообщений об ошибках для соответствующих PR имеется категория `java`, она относится к работе над портированием JDK, которые проводит Проект FreeBSD Java. Таким образом, вы должны относить свой Java-порт, как и любой другой, к категории `ports`, если решаемый вами вопрос не относится ни к реализации JDK, ни к bsd.java.mk.

Похожим образом определена политика по отношению к `CATEGORIES` порта Java, которая подробно описана в Разделение по категориям.

6.13. Веб-приложения, Apache и PHP

6.13.1. Apache

Таблица 21. Переменные для портов, использующих Apache

`USE_APACHE`	Порт требует Apache. Возможные значения: `yes` (берёт любую версию), `22`, `24`, `22-24`, `22+` и так далее. Версия по умолчанию `22`. Более подробная информация содержится в файле ports/Mk/bsd.apache.mk и на странице wiki.freebsd.org/Apache/.
`APXS`	Полный путь к исполняемому файлу `apxs`. Может быть переопределен в вашем порту.

HTTPD	Полный путь к исполняемому файлу httpd. Может быть переопределен в вашем порту.
APACHE_VERSION	Версия установленного Apache (переменная только для чтения). Эта переменная доступна только после подключения bsd.port.pre.mk. Возможные значения: 22, 24.
APACHEMODDIR	Каталог для модулей Apache. Значение переменной автоматически подставляется в pkg-plist.
APACHEINCLUDEDIR	Каталог для заголовков Apache. Значение переменной автоматически подставляется в pkg-plist.
APACHEETCDIR	Каталог для конфигурационных файлов Apache. Значение переменной автоматически подставляется в pkg-plist.

Таблица 22. Используемые переменные при портировании модулей Apache

MODULENAME	Название модуля. Значением по умолчанию является PORTNAME. Пример: mod_hello
SHORTMODNAME	Краткое название модуля. Наследуется автоматически от MODULENAME, но может быть переопределено. Пример: hello
AP_FAST_BUILD	Использовать apxs для компиляции и установки модуля.
AP_GENPLIST	Также автоматически создает pkg-plist.
AP_INC	Добавляет каталог к пути поиска заголовков во время компиляции.
AP_LIB	Добавляет каталог к пути поиска библиотек во время компиляции.
AP_EXTRAS	Дополнительные флаги, передаваемые apxs.

6.13.2. Веб-приложения

Веб-приложения следует устанавливать в PREFIX/www/appname. Для вашего удобства этот путь одинаково доступен в Makefile и pkg-plist как переменная WWWDIR, а путь относительно PREFIX доступен в Makefile как WWWDIR_REL.

Пользователь и группа процесса веб-сервера доступны как WWWOWN и WWWGRP, в случае если вам нужно изменить владельца для некоторых файлов. Значением по умолчанию и для владельца, и для группы является www. Если вы хотите использовать в вашем порте другие значения, воспользуйтесь для этого нотацией WWWOWN?= myuser, чтобы позволить пользователю легко переопределить их.

Не добавляйте зависимость от Apache, если веб-приложение явным образом не нуждается в

Apache. Учитывайте, что пользователи могут пожелать запустить ваше веб-приложение на другом веб-сервере помимо Apache.

6.13.3. PHP

Таблица 23. Переменные для портов, использующих PHP

USE_PHP	Порт требует PHP. Значение yes добавляет зависимость от PHP. Вместо этого может быть указан перечень требуемых расширений PHP. Пример: pcre xml gettext
DEFAULT_PHP_VER	Выбирает старший номер версии, с которым будет установлен PHP как зависимость в случае, когда PHP еще не установлен. По умолчанию 5. Возможные значения: 4, 5
IGNORE_WITH_PHP	Порт не работает с PHP данной версии. Возможные значения: 4, 5
USE_PHPIZE	Порт будет построен как расширение PHP.
USE_PHPEXT	Порт будет считаться расширением PHP, включая установку и регистрацию в реестре расширений.
USE_PHP_BUILD	Установить PHP как зависимость времени построения.
WANT_PHP_CLI	Хочет CLI (командная строка) версию PHP.
WANT_PHP_CGI	Хочет CGI версию PHP.
WANT_PHP_MOD	Хочет PHP как модуль Apache.
WANT_PHP_SCR	Хочет CLI или CGI версию PHP.
WANT_PHP_WEB	Хочет модуль Apache или CGI версию PHP.

6.13.4. Модули PEAR

Портирование модулей PEAR является очень простым процессом.

Используйте переменные FILES, TESTS, DATA, SQLS, SCRIPTFILES, DOCS and EXAMPLES для перечисления файлов, которые вы хотите установить. Все перечисленные файлы будут автоматически установлены в подходящие места и добавлены в pkg-plist.

Подключите ${PORTSDIR}/devel/pear/bsd.pear.mk на последней строке Makefile.

Пример 21. Пример Makefile для классов PEAR

```
PORTNAME=   Date
PORTVERSION=    1.4.3
CATEGORIES= devel www pear
```

```
MAINTAINER= example@domain.com
COMMENT=    PEAR Date and Time Zone Classes

BUILD_DEPENDS=  ${PEARDIR}/PEAR.php:${PORTSDIR}/devel/pear-PEAR
RUN_DEPENDS:=   ${BUILD_DEPENDS}

FILES=      Date.php Date/Calc.php Date/Human.php Date/Span.php     \
        Date/TimeZone.php
TESTS=      test_calc.php test_date_methods_span.php testunit.php   \
        testunit_date.php testunit_date_span.php wknotest.txt   \
        bug674.php bug727_1.php bug727_2.php bug727_3.php       \
        bug727_4.php bug967.php weeksinmonth_4_monday.txt       \
        weeksinmonth_4_sunday.txt weeksinmonth_rdm_monday.txt   \
        weeksinmonth_rdm_sunday.txt
DOCS=       TODO
_DOCSDIR=   .

.include <bsd.port.pre.mk>
.include "${PORTSDIR}/devel/pear/bsd.pear.mk"
.include <bsd.port.post.mk>
```

6.14. Использование Python

Коллекция Портов поддерживает параллельную установку множества версий Python. Следует убедиться, что в портах используется правильный интерпретатор `python` в соответствии с переменной `PYTHON_VERSION`, установленной пользователем. По большей части это означает замену пути к исполняемому файлу `python` в сценариях на значение переменной `PYTHON_CMD`.

Порты, устанавливающие файлы под каталог `PYTHON_SITELIBDIR`, должны использовать префикс вида `pyXY-`, таким образом названия пакетов будут включать в себя версию Python, с которой они установлены.

```
PKGNAMEPREFIX=  ${PYTHON_PKGNAMEPREFIX}
```

Таблица 24. Переменные для портов, которые используют Python

`USE_PYTHON`	Для этого порта нужен Python. Минимальная требуемая версия может быть указана с таким значением как `2.6+`. Также можно указан диапазон версий с разделением двух версий через -, например: `2.6-2.7`

USE_PYDISTUTILS	Использовать дистрибутивные утилиты (distutils) Python для конфигурации, компиляции и установки. Необходимо, если порт использует setup.py. Переопределяет цели do-build и do-install и также может переопределять do-configure, если не определена GNU_CONFIGURE.
PYTHON_PKGNAMEPREFIX	Используется как PKGNAMEPREFIX для отличия пакетов, использующих разные версии Python. Пример: py24-
PYTHON_SITELIBDIR	Местонахождение дерева site-packages, которое содержит путь установки Python (обычно, LOCALBASE). Переменная PYTHON_SITELIBDIR может быть очень полезной при установке модулей Python.
PYTHONPREFIX_SITELIBDIR	Вариант PYTHON_SITELIBDIR без PREFIX. По возможности всегда используйте %%PYTHON_SITELIBDIR%% в pkg-plist. Значением по умолчанию для %%PYTHON_SITELIBDIR%% является lib/python%%PYTHON_VERSION%%/site-packages.
PYTHON_CMD	Командная строка интерпретатора Python, включая номер версии.
PYNUMERIC	Строка зависимости для расширения numeric.
PYNUMPY	Строка зависимости для нового расширения numeric, numpy (PYNUMERIC объявлен устаревшим вышестоящим производителем).
PYXML	Строка зависимости для расширения XML (не нужно для Python 2.0 и выше, т.к. включено в основной дистрибутив).

Полный перечень доступных переменных можно найти в /usr/ports/Mk/bsd.python.mk.

Некоторые приложения на Python заявляют о поддержке DESTDIR (требуется для staging), которая не работает (в частности, у Mailman до версии 2.1.16). Ограничение можно обойти путём перекомпиляции сценариев. Например, это можно выполнить в цели post-build. С учётом того, что после установки предполагаемое место размещения сценариев Python будет находиться в PYTHONPREFIX_SITELIBDIR, можно применить следующее решение:

```
(cd ${STAGEDIR}${PREFIX} \
  && ${PYTHON_CMD} ${PYTHON_LIBDIR}/compileall.py \
   -d ${PREFIX} -f ${PYTHONPREFIX_SITELIBDIR:S;${PREFIX}/;;})
```

Эта команда перекомпилирует исходный текст с заменой путей на относительные к каталогу сборки, а также дописывает значение PREFIX перед именем файла, записанного в выходном файле с промежуточным представлением, с использованием -d. -f требуется для безусловной перекомпиляции, :S;${PREFIX}/;; удаляет префиксы из значения переменной PYTHONPREFIX_SITELIBDIR, чтобы сделать его относительным к PREFIX.

Для этого требуется Python 2.7 или выше. Это не работает с Python 2.6.

6.15. Использование Tcl/Tk

В Коллекции Портов поддерживается одновременная установка множественных версий Tcl/Tk. Порты должны пытаться поддерживать по крайней мере версию Tcl/Tk, используемую по умолчанию, и выше с помощью переменных USE_TCL и USE_TK. Желаемую версию tcl можно указать в переменной WITH_TCL_VER.

Таблица 25. Наиболее востребованные переменные для портов, которые используют Tcl/Tk

USE_TCL	Порт зависит от библиотеки Tcl (не оболочки). Минимальную требуемую версию можно указать с использованием таких значений, как 84+. Отдельные неподдерживаемые версии указываются в переменной INVALID_TCL_VER.
USE_TCL_BUILD	Tcl нужен для порта только на время сборки.
USE_TCL_WRAPPER	Эту новую переменную следует использовать для портов, для которых требуется оболочка Tcl и не требуется конкретная версия tclsh. Обертка tclsh устанавливается в систему. Пользователь может указать желаемую оболочку tcl для использования.
WITH_TCL_VER	Определяемые пользователем переменные, которые устанавливают желаемую версию Tcl.
UNIQUENAME_WITH_TCL_VER	Подобно WITH_TCL_VER, но для каждого порта.
USE_TCL_THREADS	Требует многопоточную сборку Tcl/Tk.
USE_TK	Порт зависит от библиотеки Tk (не от предпочитаемой оболочки). Подразумевает USE_TCL с тем же значением. Для большей информации смотрите описание переменной USE_TCL.
USE_TK_BUILD	Аналогично USE_TCL_BUILD.
USE_TK_WRAPPER	Аналогично USE_TCL_WRAPPER.
WITH_TK_VER	Аналогично WITH_TCL_VER, подразумевает WITH_TCL_VER той же версии.

Полный перечень доступных переменных находится в /usr/ports/Mk/bsd.tcl.mk.

6.16. Использование Emacs

Этот раздел ещё предстоит написать.

6.17. Использование Ruby

Таблица 26. Полезные переменные для портов, использующих Ruby

Переменная	Описание
`USE_RUBY`	Порт требует Ruby.
`USE_RUBY_EXTCONF`	Порт использует для конфигурации extconf.rb.
`USE_RUBY_SETUP`	Порт использует для конфигурации setup.rb.
`RUBY_SETUP`	Устанавливает альтернативное имя для setup.rb. Распространенным значением является install.rb.

Следующая таблица отражает некоторые переменные, доступные авторам портов через инфраструктуру портов. Эти переменные должны использоваться для установки файлов в правильное месторасположение. Используйте их в pkg-plist как можно больше. Эти переменные не должны переопределяться в самом порте.

Таблица 27. Отобранные переменные только для чтения для портов, использующих Ruby

Переменная	Описание	Примерное значение
`RUBY_PKGNAMEPREFIX`	Используется как `PKGNAMEPREFIX` для различия пакетов от разных версий Ruby.	`ruby18-`
`RUBY_VERSION`	Полная версия Ruby в форме `x.y.z`.	`1.8.2`
`RUBY_SITELIBDIR`	Путь для установки архитектуронезависимых библиотек.	`/usr/local/lib/ruby/site_ruby/1.8`
`RUBY_SITEARCHLIBDIR`	Путь для установки архитектурозависимых библиотек.	`/usr/local/lib/ruby/site_ruby/1.8/amd64-freebsd6`
`RUBY_MODDOCDIR`	Путь для установки документации модуля.	`/usr/local/shared/doc/ruby18/patsy`
`RUBY_MODEXAMPLESDIR`	Путь для установки примеров модуля.	`/usr/local/shared/examples/ruby18/patsy`

Полный перечень доступных переменных находится в /usr/ports/Mk/bsd.ruby.mk.

6.18. Использование SDL

Переменная `USE_SDL` используется для автоматической настройки зависимостей для портов, использующих библиотеки на основе SDL, такие как devel/sdl12 или graphics/sdl_image.

Для версии 1.2 на данный момент распознаются следующие SDL-библиотеки:

- sdl: devel/sdl12
- console: devel/sdl_console
- gfx: graphics/sdl_gfx
- image: graphics/sdl_image
- mixer: audio/sdl_mixer
- mm: devel/sdlmm
- net: net/sdl_net
- pango: x11-toolkits/sdl_pango
- sound: audio/sdl_sound
- ttf: graphics/sdl_ttf

Для версии 2.0 на данный момент распознаются следующие SDL-библиотеки:

- sdl: devel/sdl20
- gfx: graphics/sdl2_gfx
- image: graphics/sdl2_image
- mixer: audio/sdl2_mixer
- net: net/sdl2_net
- ttf: graphics/sdl2_ttf

Таким образом, если порт имеет зависимость от net/sdl_net и audio/sdl_mixer, то строка будет следующей:

```
USE_SDL=    net mixer
```

Зависимость от порта devel/sdl12, который требуется для net/sdl_net и audio/sdl_mixer, будет также автоматически добавлен.

Если вы используете `USE_SDL` с элементами SDL 1.2, то он автоматически:

- Добавляет зависимость от sdl12-config к `BUILD_DEPENDS`
- Добавляет переменную `SDL_CONFIG` к `CONFIGURE_ENV`
- Добавляет зависимости от указанных библиотек к `LIB_DEPENDS`

Если вы используете `USE_SDL` с элементами SDL 2.0, то он автоматически:

- Добавляет зависимость от sdl2-config к `BUILD_DEPENDS`
- Добавляет переменную `SDL2_CONFIG` к `CONFIGURE_ENV`
- Добавляет зависимости от указанных библиотек к `LIB_DEPENDS`

Для проверки наличия библиотеки SDL вы можете делать это при помощи переменной `WANT_SDL`:

```
WANT_SDL=   yes

.include <bsd.port.pre.mk>

.if ${HAVE_SDL:Mmixer}!=""
USE_SDL+=   mixer
.endif

.include <bsd.port.post.mk>
```

6.19. Использование wxWidgets

Эта глава описывает статус библиотек wxWidgets в дереве портов и их интеграцию с системой портов.

6.19.1. Введение

Существует множество версий библиотек wxWidgets, конфликтующих между собой (устанавливают файлы под тем же именем). В дереве портов эта проблема решена путем установки каждой версии под собственным названием с использованием номера версии в качестве суффикса.

Очевидным недостатком этого является необходимость изменения каждого приложения для нахождения искомой версии. К счастью, большинство приложений для определения нужного компилятора и флагов компоновки вызывают сценарий `wx-config`. Для каждой доступной версии этот сценарий имеет своё имя. Большинство приложений учитывают переменную окружения или принимают аргумент configure для указания, какой сценарий `wx-config` следует вызывать. На все остальные приходится накладывать патч.

6.19.2. Выбор версии

Для того, чтобы заставить ваш порт использовать конкретную версию wxWidgets, существует две доступные для определения переменные (если определена только одна, то вторая примет значение по умолчанию):

Таблица 28. Переменные для выбора версии wxWidgets

Переменная	Описание	Значение по умолчанию
`USE_WX`	Перечень версий, которые порт может использовать	Все доступные версии

Переменная	Описание	Значение по умолчанию
USE_WX_NOT	Перечень версий, которые порт не может использовать	Нет

Перечень доступных версий wxWidgets и соответствующих им портов в дереве:

Таблица 29. Доступные версии wxWidgets

Версия	Порт
2.4	x11-toolkits/wxgtk24
2.6	x11-toolkits/wxgtk26
2.8	x11-toolkits/wxgtk28

Версии начиная с 2.5 также поставляются с Unicode и устанавливается подчиненным портом с названием как как у обычного, но с суффиксом -unicode, но этим можно управлять при помощи переменных (смотрите Unicode).

Переменные в Переменные для выбора версии wxWidgets можно установить в одну или более следующих комбинаций, разделенных пробелами:

Таблица 30. Определение версии для wxWidgets

Описание	Пример
Единичная версия	2.4
Восходящий диапазон	2.4+
Нисходящий диапазон	2.6-
Полный диапазон (обязан быть восходящим)	2.4-2.6

Кроме того, существует несколько переменных для выбора предпочитаемых версий из перечня доступных. Они могут быть установлены в несколько версий, первая из которых будет иметь наибольший приоритет.

Таблица 31. Переменные для выбора предпочитаемых версий wxWidgets

Название	Предназначение
WANT_WX_VER	порт
WITH_WX_VER	пользователь

6.19.3. Выбор компонентов

Существуют другие приложения, которые, хотя и не являются библиотеками wxWidgets, но в тоже время относятся к ним. Эти приложения можно указать в переменной WX_COMPS. Доступны следующие компоненты:

Таблица 32. Доступные компоненты wxWidgets

Название	Описание	Ограничение версии
`wx`	основная библиотека	нет
`contrib`	сторонние библиотеки	`нет`
`python`	wxPython (привязки к Python)	`2.4-2.6`
`mozilla`	wxMozilla	`2.4`
`svg`	wxSVG	`2.6`

Тип добавляемой зависимости при выборе каждого компонента может быть указан вручную путем добавления суффикса, отделенного точкой с запятой. Если таковой отсутствует, но будет использовано значение по умолчанию (смотрите Типы зависимости wxWidgets, используемые по умолчанию). Доступные типы зависимости:

Таблица 33. Доступные типы зависимости wxWidgets

Название	Описание
`build`	Компонент требуется для построения, эквивалентен `BUILD_DEPENDS`
`run`	Компонент требуется для запуска, эквивалентен `RUN_DEPENDS`
`lib`	Компонент требуется для построения и запуска, эквивалентен `LIB_DEPENDS`

Значения по умолчанию для компонентов подробно рассматриваются в следующей таблице:

Таблица 34. Типы зависимости wxWidgets, используемые по умолчанию

Компонент	Тип зависимости
`wx`	`lib`
`contrib`	`lib`
`python`	`run`
`mozilla`	`lib`
`svg`	`lib`

Пример 22. Выбор компонентов wxWidgets

Следующий фрагмент относится к порту, в котором используется wxWidgets версии `2.4` с его сторонними библиотеками.

```
USE_WX=     2.4
WX_COMPS=   wx contrib
```

6.19.4. Unicode

Библиотека wxWidgets поддерживает Unicode начиная с версии 2.5. В дереве портов доступны обе версии и могут быть выбраны с использованием следующих переменных:

Таблица 35. Переменные для выбора версии wxWidgets с Unicode

Переменная	Описание	Предназначение
WX_UNICODE	Порт работает *только* с версией Unicode	порт
WANT_UNICODE	Порт работает с обеими версиями, но предпочитает версию с Unicode	порт
WITH_UNICODE	Порт будет использовать версию Unicode	пользователь
WITHOUT_UNICODE	Порт будет использовать обычную версию, если это поддерживается (когда WX_UNICODE не определена)	пользователь

Не используйте WX_UNICODE для портов, которые могут использовать обе версии. Если вы хотите, чтобы порт по умолчанию использовал Unicode, определите вместо этого WANT_UNICODE.

6.19.5. Обнаружение установленных версий

Для обнаружения установленной версии вам необходимо задать переменную WANT_WX. Если вы не присвоите ей определенную версию, то компоненты получат суффикс версии. Переменная HAVE_WX будет заполнена после обнаружения.

Пример 23. Обнаружение установленных версий и компонентов wxWidgets

Следующий фрагмент может быть использован в порту, который использует wxWidgets, в случае если он установлен или выбран соответствующий параметр.

```
WANT_WX=    yes

.include <bsd.port.pre.mk>

.if defined(WITH_WX) || !empty(PORT_OPTIONS:MWX) || !empty(HAVE_WX:Mwx-2.4)
USE_WX=         2.4
CONFIGURE_ARGS+=    --enable-wx
.endif
```

Следующий фрагмент может быть использован в порту, который задействует поддержку wxPython, в случае если он установлен или выбран соответствующий параметр, в дополнение к wxWidgets, обе версии 2.6.

```
USE_WX=     2.6
WX_COMPS=   wx
WANT_WX=    2.6

.include <bsd.port.pre.mk>

.if defined(WITH_WXPYTHON) || !empty(PORT_OPTIONS:MWXPYTHON) ||
!empty(HAVE_WX:Mpython)
WX_COMPS+=      python
CONFIGURE_ARGS+=    --enable-wxpython
.endif
```

6.19.6. Переменные для определения

Следующие переменные доступны в порту (после определения одной из переменных из Переменные для выбора версии wxWidgets).

Таблица 36. Переменные, определенные для портов, использующих wxWidgets

Название	Описание
`WX_CONFIG`	Путь к сценарию wxWidgets `wx-config` (с другим именем)
`WXRC_CMD`	Путь к программе wxWidgets `wxrc` (с другим именем)
`WX_VERSION`	Версия wxWidgets, которая будет использоваться (например, `2.6`)
`WX_UNICODE`	Если не определена, но Unicode будет использоваться, то она будет определена

6.19.7. Обработка в bsd.port.pre.mk

Если вам нужно использовать переменные для запуска команд сразу после подключения bsd.port.pre.mk, то вам нужно определить `WX_PREMK`.

Если вы определите `WX_PREMK`, то версия, зависимости, компоненты и заданные переменные не изменяться, в случае вы изменили переменные порта wxWidgets после подключения bsd.port.pre.mk.

Пример 24. Использование переменных wxWidgets в командах

Следующий фрагмент иллюстрирует использование переменной `WX_PREMK` посредством запуска сценария `wx-config` для получения строки с полной версией с присвоением ее переменной и передачей в программу.

```
USE_WX=     2.4
```

```
WX_PREMK=   yes

.include <bsd.port.pre.mk>

.if exists(${WX_CONFIG})
VER_STR!=   ${WX_CONFIG} --release

PLIST_SUB+= VERSION="${VER_STR}"
.endif
```

Переменные wxWidgets можно безопасно использовать в командах внутри целей без необходимости в использовании `WX_PREMK`.

6.19.8. Дополнительные параметры `configure`

Некоторые сценарии GNU `configure` не могут найти wxWidgets только с установленной переменной окружения `WX_CONFIG`, требуя дополнительные параметры. Для их передачи можно использовать переменную `WX_CONF_ARGS`.

Таблица 37. Допустимые значения `WX_CONF_ARGS`

Возможное значение	Получаемый параметр
`absolute`	`--with-wx-config=${WX_CONFIG}`
`relative`	`--with-wx=${LOCALBASE} --with-wx-config=${WX_CONFIG:T}`

6.20. Использование Lua

Эта глава описывает статус библиотек Lua в дереве портов и их интеграцию в систему портов.

6.20.1. Введение

Существует множество версий библиотек Lua и соответствующих интерпретаторов, конфликтующих между собой (устанавливают файлы под тем же именем). В дереве портов эта проблема решена путем установки каждой версии в собственное место с использованием номера версии в качестве суффикса.

Очевидным недостатком этого является необходимость изменения каждого приложения для нахождения искомой версии. Но это решается добавлением некоторых дополнительных флагов для компилятора и компоновщика.

6.20.2. Выбор версии

Для того, чтобы заставить ваш порт использовать конкретную версию Lua, существует две доступные для определения переменные (если определена только одна, то вторая примет значение по умолчанию):

Таблица 38. Переменные для выбора версии Lua

Переменная	Описание	Значение по умолчанию
`USE_LUA`	Перечень версий, которые порт может использовать	Все доступные версии
`USE_LUA_NOT`	Перечень версий, которые порт не может использовать	Пусто

Перечень доступных версий Lua и соответствующих портов в дереве:

Таблица 39. Доступные версии Lua

Версия	Порт
`4.0`	lang/lua4
`5.0`	lang/lua50
`5.1`	lang/lua

Переменные из Переменные для выбора версии Lua могут иметь комбинации из одного или нескольких значений, разделенных пробелом:

Таблица 40. Определение версии Lua

Описание	Пример
Единичная версия	`4.0`
Восходящий диапазон	`5.0+`
Нисходящий диапазон	`5.0-`
Полный диапазон (обязан быть восходящим)	`5.0-5.1`

Кроме того, существует несколько переменных для выбора предпочитаемых версий из перечня доступных. Они могут быть установлены в несколько версий, первая из которых будет иметь наибольший приоритет.

Таблица 41. Переменные для выбора предпочитаемых версий Lua

Название	Предназначение
`WANT_LUA_VER`	порт
`WITH_LUA_VER`	пользователь

Пример 25. Выбор версии Lua

Следующий фрагмент взят из порта, который использует Lua версий `5.0` или `5.1`, по умолчанию `5.0`. Значение может быть переопределено пользователем с использованием переменной `WITH_LUA_VER`.

```
USE_LUA=    5.0-5.1
WANT_LUA_VER=   5.0
```

6.20.3. Выбор компонентов

Существуют другие приложения, которые хотя и не являются библиотеками Lua, но относятся к ним. Эти приложения можно указать в переменной `LUA_COMPS`. Доступны следующие компоненты:

Таблица 42. Доступные компоненты Lua

Название	Описание	Ограничение версии
`lua`	Основная библиотека	нет
`tolua`	Библиотека доступа к коду C/C++	4.0-5.0
`ruby`	Привязка к Ruby	4.0-5.0

Есть и другие компоненты, но они относятся к модулям для интерпретатора и не используются приложениями (только другими модулями).

Тип зависимости можно выбрать для каждого компонента через добавление суффикса, отделенного точкой с запятой. В случае отсутствия будет использован тип по умолчанию (смотрите Типы зависимости Lua, используемые по умолчанию). Доступные следующие типы:

Таблица 43. Доступные типы зависимости Lua

Название	Описание
`build`	Компонент требуется для построения, эквивалентен `BUILD_DEPENDS`
`run`	Компонент требуется для запуска, эквивалентен `RUN_DEPENDS`
`lib`	Компонент требуется для построения и запуска, эквивалентен `LIB_DEPENDS`

Значения по умолчанию для компонентов подробно рассматриваются в следующей таблице:

Таблица 44. Типы зависимости Lua, используемые по умолчанию

Компонент	Тип зависимости
`lua`	`lib` для `4.0-5.0` (динамическая) и `build` для `5.1` (статическая)
`tolua`	`build` (статическая)
`ruby`	`lib` (динамическая)

Пример 26. Выбор компонентов Lua

Следующий фрагмент соответствует порту, использующему Lua версии `4.0` и привязку к Ruby.

```
USE_LUA=    4.0
LUA_COMPS=  lua ruby
```

6.20.4. Обнаружение установленных версий

Для обнаружения установленной версии вам необходимо задать переменную `WANT_LUA`. Если вы не присвоите ей определенную версию, то компоненты получат суффикс версии. Переменная `HAVE_LUA` будет заполнена после обнаружения.

Пример 27. Обнаружение установленных версий и компонентов Lua

Следующий фрагмент можно использовать для порта, использующего Lua, если она установлена, или был выбран соответствующий параметр.

```
WANT_LUA=   yes

.include <bsd.port.pre.mk>

.if defined(WITH_LUA5) || !empty(PORT_OPTIONS:MLUA5) || !empty(HAVE_LUA:Mlua-
5.[01])
USE_LUA=        5.0-5.1
CONFIGURE_ARGS+=    --enable-lua5
.endif
```

Следующий фрагмент можно использовать для порта, который включает поддержку tolua, если такой компонент установлен, или был выбран соответствующий параметр в дополнение к Lua, оба имеют версию `4.0`.

```
USE_LUA=    4.0
LUA_COMPS=  lua
WANT_LUA=   4.0

.include <bsd.port.pre.mk>

.if defined(WITH_TOLUA) || !empty(PORT_OPTIONS:MTOLUA) || !empty(HAVE_LUA:Mtolua)
LUA_COMPS+=     tolua
CONFIGURE_ARGS+=    --enable-tolua
.endif
```

6.20.5. Переменные для определения

Следующие переменные доступны в порту (после определения одной из переменных из Переменные для выбора версии Lua).

Таблица 45. Переменные, определенные для портов, использующих Lua

Название	Описание
LUA_VER	Версия Lua, которая будет использоваться (например, 5.1)
LUA_VER_SH	Старший номер версии динамической библиотеки Lua (например, 1)
LUA_VER_STR	Версия Lua без точки (например, 51)
LUA_PREFIX	Префикс, в который установлена Lua (и компоненты)
LUA_SUBDIR	Каталог под ${PREFIX}/bin, ${PREFIX}/share и ${PREFIX}/lib, в который установлена Lua
LUA_INCDIR	Каталог, в который установлены заголовочные файлы Lua и tolua
LUA_LIBDIR	Каталог, в который установлены библиотеки Lua и tolua
LUA_MODLIBDIR	Каталог, в который установлены модули библиотеки Lua (.so)
LUA_MODSHAREDIR	Каталог, в который установлены модули Lua (.lua)
LUA_PKGNAMEPREFIX	Префикс с именем пакета, используемый модулями Lua
LUA_CMD	Путь к интерпретатору Lua
LUAC_CMD	Путь к компилятору Lua
TOLUA_CMD	Путь к программе tolua

Пример 28. Указание для порта, где искать Lua

Следующий фрагмент показывает, как сообщить порту, который использует сценарий configure, где расположены заголовочные файлы и библиотеки Lua.

```
USE_LUA=    4.0
GNU_CONFIGURE=  yes
CONFIGURE_ENV=  CPPFLAGS="-I${LUA_INCDIR}" LDFLAGS="-L${LUA_LIBDIR}"
```

6.20.6. Обработка в bsd.port.pre.mk

Если вам нужно использовать переменные для запуска команд сразу после подключения bsd.port.pre.mk, для этого вам нужно определить переменную LUA_PREMK.

Если вы задаете LUA_PREMK, то версия, зависимости, компоненты и уже заданные переменные не будут изменены, в случае если вы изменили переменные порта Lua после подключения bsd.port.pre.mk.

Пример 29. Использование переменных Lua в командах

Следующий фрагмент иллюстрирует использование LUA_PREMK посредством запуска интерпретатора Lua для того, чтобы получить строку с полной версией, сохранить ее в переменную и передать программе.

```
USE_LUA=    5.0
LUA_PREMK=  yes

.include <bsd.port.pre.mk>

.if exists(${LUA_CMD})
VER_STR!=   ${LUA_CMD} -v

CFLAGS+=    -DLUA_VERSION_STRING="${VER_STR}"
.endif
```

Переменные Lua можно безопасно использовать в командах внутри целей без необходимости в использовании LUA_PREMK.

6.21. Использование iconv

После 10-08-2013 (r254273) в составе FreeBSD 10-CURRENT и более новых версий имеется собственный iconv. В более ранних версиях дополнительной зависимостью выступал converters/libiconv.

Для программного обеспечения, которому нужен iconv, определите USES=iconv. Версии FreeBSD до 10-CURRENT от 13-08-2013 (r254273) не имеют собственного iconv. На этих более ранных версиях будет автоматически добавлена зависимость от converters/libiconv.

Когда порт задаёт USES=iconv, становятся доступными следующие переменные:

Имя переменной	Назначение	Значение до FreeBSD 10-CURRENT 254273 (13-08-2013)	Значение после FreeBSD 10-CURRENT 254273 (13-08-2013)
ICONV_CMD	Каталог размещения двоичного файла iconv	${LOCALBASE}/bin/iconv	/usr/bin/iconv
ICONV_LIB	Аргумент ld для компоновки с libiconv (если нужно)	-liconv	(пусто)

Имя переменной	Назначение	Значение до FreeBSD 10-CURRENT 254273 (13-08-2013)	Значение после FreeBSD 10-CURRENT 254273 (13-08-2013)
ICONV_PREFIX	Каталог размещения реализации iconv (используется для сценариев конфигурации)	${LOCALBASE}	/usr
ICONV_CONFIGURE_ARG	Параметр предварительно собранной конфигурации для сценариев конфигурации	--with-libiconv -prefix=${LOCALBASE}	(пусто)
ICONV_CONFIGURE_BASE	Параметр предварительно собранной конфигурации для сценариев конфигурации	--with -libiconv=${LOCALBASE}	(пусто)

В следующих двух примерах демонстрируется автоматическое присвоение переменным правильных значений для систем, использующих converters/libiconv или собственный iconv.

Пример 30. Простое использование iconv

```
USES=        iconv
LDFLAGS+=    -L${LOCALBASE}/lib ${ICONV_LIB}
```

Пример 31. Использование iconv с configure

```
USES=        iconv
CONFIGURE_ARGS+=${ICONV_CONFIGURE_ARG}
```

Как показано выше, ICONV_LIB имеет пустое значение с собственным iconv. Эту особенность можно использовать для обнаружения собственного iconv с соответствующими действиями.

Иногда в программе параметр ld или путь поиска жёстко заданы в Makefile или сценарии конфигурации. Для решения этой проблемы можно использовать следующий подход:

Пример 32. Исправление жёстко заданного -liconv

```
USES=        iconv
```

```
post-patch:
    @${REINPLACE_CMD} -e 's/-liconv/${ICONV_LIB}/' ${WRKSRC}/Makefile
```

В некоторых случаях необходимо установить альтернативные значения или выполнить операции в случае использования собственного `iconv`. Перед проверкой значения `ICONV_LIB` обязан быть подключён bsd.port.pre.mk:

Пример 33. Проверка доступности собственного `iconv`

```
USES=       iconv

.include <bsd.port.pre.mk>

post-patch:
.if empty(ICONV_LIB)
    # обнаружен собственный iconv
    @${REINPLACE_CMD} -e 's|iconv||' ${WRKSRC}/Config.sh
.endif

.include <bsd.port.post.mk>
```

6.22. Использование Xfce

Переменная `USE_XFCE` используется для автоматической конфигурации зависимостей для портов, использующих библиотеки или приложения на основе Xfce, такие как x11-toolkits/libxfce4gui и x11-wm/xfce4-panel.

В настоящее время распознаются следующие библиотеки и приложения Xfce:

- libexo: x11/libexo
- libgui: x11-toolkits/libxfce4gui
- libutil: x11/libxfce4util
- libmcs: x11/libxfce4mcs
- mcsmanager: sysutils/xfce4-mcs-manager
- panel: x11-wm/xfce4-panel
- thunar: x11-fm/thunar
- wm: x11-wm/xfce4-wm
- xfdev: dev/xfce4-dev-tools

Распознаются следующие дополнительные параметры:

- configenv: Используйте, если ваш порт требует специально измененного значения

CONFIGURE_ENV для поиска требуемых для порта библиотек.

```
-I${LOCALBASE}/include -L${LOCALBASE}/lib
```

добавляется в CPPFLAGS к CONFIGURE_ENV.

Следовательно, если у порта имеется зависимость от sysutils/xfce4-mcs-manager, и порт требует специальных CPPFLAGS в своем окружении configure, то синтаксис будет следующим:

```
USE_XFCE=   mcsmanager configenv
```

6.23. Использование Mozilla

Таблица 46. Переменные для портов, использующих Mozilla

USE_GECKO	Один из бэкэндов Gecko, с которым может работать порт. Возможные значения: libxul (libxul.so), seamonkey (libgtkembedmoz.so, устаревший, больше не должен использоваться).
USE_FIREFOX	Для запуска порта требуется Firefox. Возможные значения: yes (версия по умолчанию), 40, 36, 35. По умолчанию устанавливает зависимость от версии 40.
USE_FIREFOX_BUILD	Для построения порта требуется Firefox. Возможные значения: смотрите USE_FIREFOX. Автоматически устанавливает USE_FIREFOX с присвоением того же значения.
USE_SEAMONKEY	Для запуска порта требуется SeaMonkey. Возможные значения: yes (версия по умолчанию), 20, 11 (устарело, больше не должно использоваться). По умолчанию устанавливает зависимость от версии 20.
USE_SEAMONKEY_BUILD	Для построения порта требуется SeaMonkey. Возможные значения: смотрите USE_SEAMONKEY. Автоматически устанавливает USE_SEAMONKEY с присвоением того же значения.

USE_THUNDERBIRD	Для запуска порта требуется Thunderbird. Возможные значения: yes (версия по умолчанию), 31, 30 (устарело, больше не должно использоваться). По умолчанию устанавливает зависимость от версии 31.
USE_THUNDERBIRD_BUILD	Для построения порта требуется Thunderbird. Возможные значения: смотрите USE_THUNDERBIRD. Автоматически устанавливает USE_THUNDERBIRD с присвоением того же значения.

Полный перечень доступных переменных можно получить в файле /usr/ports/Mk/bsd.gecko.mk.

6.24. Использование баз данных

Таблица 47. Переменные для портов, использующих базы данных

Переменная	Значение
USE_BDB	Если переменная установлена в yes, добавляет зависимость от порта databases/db41. Также переменной можно присвоить значения: 2, 3, 40, 41, 42, 43, 44, 46, 47, 48 или 51. Вы можете объявить диапазон принимаемых значений, USE_BDB=42+ будет искать установленную версию с наибольшим номером, и, если ничего не установлено, вернется к 42.
USE_MYSQL	Если переменная установлена в yes, добавляет зависимость от порта databases/mysql55-client. Как связанная переменная, WANT_MYSQL_VER может быть установлена в значение 323, 40, 41, 50, 51, 52, 55 или 60.
USE_PGSQL	Если установлена в yes, добавляет зависимость от порта databases/postgresql90-client. Как связанная переменная, WANT_PGSQL_VER может быть установлена в значение 83, 84, 90, 91 или 92. Вы можете указать максимальное и минимальное значения; WANT_PGSQL_VER=90+ сделает порт зависимым от минимальной версии 9.0.
USE_SQLITE	Если переменная имеет значение yes, добавляет зависимость от порта databases/sqlite3. Переменная может принимать значения: 3, 2.

Подробнее смотрите в bsd.database.mk.

6.25. Запуск и остановка служб (сценарии `rc`)

Сценарии rc.d используются для запуска служб при запуске системы и дают администратору стандартный способ остановки, запуска и перезапуска службы. Порты интегрируются в системную инфраструктуру rc.d. Подробности по её использованию можно найти в главе rc.d Руководства. Подробное объяснение доступных команд находится в rc(8) и rc.subr(8). Наконец, есть статьяо практических аспектах написания сценариев rc.d.

Установить можно один или более сценариев rc.d:

```
USE_RC_SUBR=    doormand
```

Сценарии обязаны размещаться в подкаталоге files с обязательным добавлением суффикса `.in` к имени файла. Для этого файла будут использоваться стандартные расширения `SUB_LIST`. Также особенно приветствуется использование расширений `%%PREFIX%%` и `%%LOCALBASE%%`. Подробнее о `SUB_LIST` в соответствующей главе.

Начиная с FreeBSD 6.1-RELEASE локальные сценарии rc.d (включая установленные из портов) включены в общий rcorder(8) основной системы.

Пример простого сценария rc.d:

```
#!/bin/sh

# $FreeBSD$
#
# PROVIDE: doormand
# REQUIRE: LOGIN
# KEYWORD: shutdown
#
# Add the following lines to /etc/rc.conf.local or /etc/rc.conf
# to enable this service:
#
# doormand_enable (bool):   Set to NO by default.
#               Set it to YES to enable doorman.
# doormand_config (path):   Set to %%PREFIX%%/etc/doormand/doormand.cf
#               by default.

. /etc/rc.subr

name=doormand
rcvar=doormand_enable

load_rc_config $name

: ${doormand_enable:="NO"}
```

```
: ${doormand_config="%%PREFIX%%/etc/doormand/doormand.cf"}

command=%%PREFIX%%/sbin/${name}
pidfile=/var/run/${name}.pid

command_args="-p $pidfile -f $doormand_config"

run_rc_command "$1"
```

Если нет стоящей причины запускать службы раньше всех портов, сценарии должны использовать

```
REQUIRE: LOGIN
```

Если служба работает под определенным пользователем (отличным от root), то это делается принудительно. В сценарий выше включена конструкция

```
KEYWORD: shutdown
```

потому что вымышленный порт, который мы используем в качестве примера, запускает службу, и она должна корректно завершиться при выключении системы. Если сценарий не запускает постоянную службу, то это не является необходимым.

Для необязательных элементов конфигурации присвоение переменной по умолчанию в стиле "=" является более предпочтительным по сравнению со стилем ":=", используемым здесь, поскольку первый устанавливает значение по умолчанию только если переменная не установлена, а последний устанавливает её, если переменная не установлена *или* обнулена. Пользователь вполне может написать в своем файле rc.conf.local что-нибудь типа

```
doormand_flags=""
```

и тогда произойдет неуместная подстановка переменной с использованием ":=", что переопределит намерения пользователя. Переменная `_enable` является обязательной; значением по умолчанию должно быть ":".

6.25.1. Контрольный список перед внесением изменений

Перед тем, как отсылать порт со сценарием rc.d, и тем более перед его коммитом, сверьтесь со следующим контрольным списком, чтобы убедиться, что порт для этого готов.

Большинство из этих проверок умеет выполнять порт devel/rclint, но это не является заменой надлежащему просмотру.

1. Если это новый файл, заканчивается ли он на .sh? Если это так, то имя файла должно быть изменено на file.in, поскольку файлы rc.d не могут оканчиваться на

такое расширение.

2. Присутствует ли в файле тег `$FreeBSD$`?
3. Соответствуют ли друг другу имя файла (без .in), строка `PROVIDE` и `$`*name*? Имя файла, совпадающее с `PROVIDE`, упрощает отладку, особенно для проблем, связанных с rcorder(8). Соответствие имени файла и `$`*name* также упрощает понимание, какие переменные имеют отношение к сценарию в rc.conf[.local]. Последнее также является тем, что вы могли бы назвать "политикой" для всех новых сценариев, включая те, что входят в базовую систему.
4. Содержит ли строка `REQUIRE` значение LOGIN? Это условие обязательно для сценариев, работающих не из-под суперпользователя. Если сценарий запускается из-под суперпользователя, то стоит ли его запускать до `LOGIN`? Если нет, то его следует запускать после, так чтобы мы могли свободно сгруппировать локальные сценарии в той точке rcorder(8), когда почти все сценарии в базовой системе уже стартовали.
5. Запускает ли сценарий постоянную службу? Если да, то он должен иметь `KEYWORD: shutdown`.
6. Убедитесь в том, что в сценарии отсутствует `KEYWORD: FreeBSD`. Это перестало быть нужным и нежелательно уже много лет. Это также служит индикатором того, что новый сценарий был скопирован со старого, поэтому особое внимание должно быть уделено при проверке.
7. Если сценарий использует интерпретируемый язык, такой как `perl`, `python` или `ruby`, то убедитесь, что значение `command_interpreter` установлено должным образом. В противном случае

   ```
   # service name stop
   ```

   возможно будет работать неправильно. Смотрите service(8) для дополнительной информации.

8. Все ли вхождения /usr/local были заменены на `%%PREFIX%%`?
9. Идет ли присвоение переменным значений по умолчанию после `load_rc_config`?
10. Используются ли пустые строки при присвоении значений по умолчанию? Такие присвоения должны быть удалены, но перепроверьте, что эти параметры задокументированы в комментариях в начале файла.
11. Действительно ли в сценариях используются значения, присвоенные переменным?
12. Являются ли параметры по умолчанию, перечисленные в *name*`_flags`, обязательными? Если это так, то их следует поместить в `command_args`. Параметр `-d` здесь - это как красный флаг (прошу прощения за каламбур), поскольку обычно он применяется для "демонизации" процесса и поэтому на самом деле обязательный.
13. Никогда не включайте переменную *name*`_flags` в `command_args` (и наоборот; в прочем, такая ошибка встречается реже).
14. Запускает ли сценарий какой-либо код безусловно? Это нехорошо. Обычно такие

вещи могут/должны помещаться в `start_precmd`.

15. Все логические условия должны использовать функцию `checkyesno`. Не пишите самописных проверок для `[Yy][Ee][Ss]`, и так далее.
16. Если в сценарии выполняется цикл (например, ожидание чего-либо перед стартом), используется ли счетчик для завершения цикла? Мы не хотим бесконечного ожидания загрузки в случае возникновения ошибки.
17. Создает ли сценарий файлы или каталоги, которым нужны особые права доступа? Например, файл pid, который должен принадлежать пользователю, из-под которого запускается процесс. Вместо традиционных команд touch(1)/chown(8)/chmod(1) подумайте об использовании install(1) с подходящими аргументами командной строки, для того чтобы выполнить всю процедуру за один шаг.

6.26. Добавление пользователей и групп

Некоторые порты требуют в установленной системе наличие определенного пользователя. Выберите свободный UID в диапазоне от 50 до 999 и зарегистрируйте его в ports/UIDs (для пользователей) и/или в ports/GIDs (для групп). Удостоверьтесь, что не используете UID, уже используемый системой или другими портами.

Пожалуйста, включите в патч изменение для этих двух файлов, если вам требуется создать нового пользователя или группу для вашего порта.

Затем вы сможете использовать в вашем Makefile переменные `USERS` и `GROUPS`, и пользователь автоматически создастся при установке порта.

```
USERS=  pulse
GROUPS= pulse pulse-access pulse-rt
```

Текущий перечень зарезервированных UID и GID находится в ports/UIDs и ports/GIDs.

6.27. Порты, требующие наличия исходных текстов ядра

Некоторым портам (таким как загружаемые модули ядра) для компиляции нужны файлы с исходными текстами ядра. Ниже указан корректный способ определения, установлены ли они пользователем:

```
USES=   kmod
```

Кроме этой проверки, `kmod` заботится о большинстве пунктов, которые должны учитываться в этих портах.

Глава 7. Продвинутые практики pkg-plist

7.1. Изменение содержимого pkg-plist в зависимости от make-переменных

Некоторые порты, в частности, порты `p5-`, должны менять содержимое своих файлов pkg-plist в зависимости от того, с какими параметрами они были отконфигурированы (или в зависимости от версии языка `perl` в случае портов `p5-`). Чтобы облегчить этот процесс, любые вхождения ключевых слов `%%OSREL%%`, `%%PERL_VER%%` и `%%PERL_VERSION%%` в файле pkg-plist будут заменяться соответствующими значениями. Значением `%%OSREL%%` является номер версии операционной системы (например, `4.9`). `%%PERL_VERSION%%` и `%%PERL_VER%%` обозначают полный номер версии `perl` (например, `5.8.9`). Некоторые другие `%%VARS%%`, имеющие отношение к файлам документации порта, описаны в соответствующем разделе.

Если вам нужно сделать другие подстановки, вы можете указать в переменной `PLIST_SUB` список пар `VAR=VALUE`, и все вхождения `%%VAR%%` в файле pkg-plist будут заменяться на значение *VALUE*.

Например, если у вас имеется порт, который устанавливает много файлов в каталог, зависящий от версии, вы можете задать нечто типа

```
OCTAVE_VERSION= 2.0.13
PLIST_SUB=  OCTAVE_VERSION=${OCTAVE_VERSION}
```

в файле Makefile и использовать `%%OCTAVE_VERSION%%` везде, где нужно указать номер версии в файле pkg-plist. Таким образом, при обновлении порта вам не нужно будет менять десятки (а в некоторых случаях и сотни) строк в файле pkg-plist.

Если ваш порт устанавливает файлы в соответствии с установленными в порту опциями, то обычным способом управления является добавление префиксов `%%TAG%%` для строк pkg-plist с добавлением этого `TAG` в переменную `PLIST_SUB` внутри Makefile со специальным значением `@comment`, которое указывает пакетным инструментам игнорировать эти строки:

```
.if defined(WITH_X11)
PLIST_SUB+= X11=""
.else
PLIST_SUB+= X11="@comment "
.endif
```

и в самом pkg-plist:

```
%%X11%%bin/foo-gui
```

Эта подстановка будет сделана между выполнением целей `pre-install` и `do-install`,

посредством чтения файла PLIST и записью в файл TMPPLIST (по умолчанию это файл WRKDIR/.PLIST.mktmp). Так что если ваш порт строит PLIST на лету, делайте это во время или до выполнения цели `pre-install`. Кроме того, если вашему порту требуется отредактировать получающийся файл, делайте это в цели `post-install` изменением файла TMPPLIST.

Другой способ изменения списка сборки порта основан на определении значений переменных `PLIST_FILES`, `PLIST_DIRS` и `PLIST_DIRSTRY`. Каждое из них рассматривается как перечень путей для записи в TMPPLIST содержимого PLIST. Имена, перечисленные в `PLIST_FILES`, `PLIST_DIRS` и `PLIST_DIRSTRY` подвергаются подстановке `%%VAR%%`, как описано выше. За исключением этого, имена из `PLIST_FILES` будут появляться в окончательном варианте перечня сборки без изменений, тогда как `@dirrm` и `@dirrmtry` будут соответственно предшествовать именам из `PLIST_DIRS` и `PLIST_DIRSTRY`. Для того чтобы изменения вступили в силу, `PLIST_FILES`, `PLIST_DIRS` и `PLIST_DIRSTRY` должны задаваться до того, как будет записываться TMPPLIST, то есть в цели `pre-install` или ещё раньше.

7.2. Пустые каталоги

7.2.1. Очистка пустых каталогов

Заставьте ваш порты удалять пустые каталоги при удалении. Обычно это достигается добавлением строк `@dirrm` для всех каталогов, которые создаются этим портом. Вам нужно удалить подкаталоги до того, как вы сможете удалить родительские каталоги.

```
 :
lib/X11/oneko/pixmaps/cat.xpm
lib/X11/oneko/sounds/cat.au
 :
@dirrm lib/X11/oneko/pixmaps
@dirrm lib/X11/oneko/sounds
@dirrm lib/X11/oneko
```

Однако, иногда `@dirrm` будет выдавать ошибки, потому что другие порты используют тот же самый подкаталог. Вы можете использовать `@dirrmtry` для удаления только пустых каталогов без выдачи предупреждений.

```
@dirrmtry share/doc/gimp
```

Эта команда не выведет никаких сообщений об ошибках и не вызовет аварийного завершения работы `pkg delete` (см. pkg-delete(8)), даже если каталог ${PREFIX}/shared/doc/gimp не пуст из-за того, что другие порты установили сюда какие-то файлы.

7.2.2. Создание пустых каталогов

Пустым каталогам, создаваемым во время установки порта, нужно особое внимание. Они не будут созданы при установке пакета, потому что пакеты содержат только файлы, а `pkg`

`add` и `pkg install` создают для них каталоги по мере надобности. Чтобы убедиться, что пустой каталог создается при установке пакета, добавьте эту строку в pkg-plist перед соответствующей строкой `@dirrm`:

```
@exec mkdir -p %D/shared/foo/templates
```

7.3. Конфигурационные файлы

Если ваш порт устанавливает конфигурационные файлы в каталог PREFIX/etc (или куда-то еще), *не* делайте их простого перечисления в файле pkg-plist. Это приведёт к тому, что по команде `pkg delete` или при новой установке файлы, тщательно отредактированные и настроенные пользователем, будут уничтожены.

Вместо этого установите файл(ы) с примерами с расширением filename.sample. Затем скопируйте файл с примером на место настоящего файла конфигурации, если таковой ещё не существует. При деинсталляции удаляйте файл конфигурации только в том случае, если он идентичен файлу с расширением .sample. Вам нужно управлять этим в Makefile и в pkg-plist (для установки из пакета).

Пример части Makefile:

```
post-install:
    @if [ ! -f ${PREFIX}/etc/orbit.conf ]; then \
    ${CP} -p ${PREFIX}/etc/orbit.conf.sample ${STAGEDIR}${PREFIX}/etc/orbit.conf ; \
    fi
```

Добавьте по три строки в pkg-plist для каждого конфигурационного файла, как показано ниже:

```
@unexec if cmp -s %D/etc/orbit.conf.sample %D/etc/orbit.conf; then rm -f
%D/etc/orbit.conf; fi
etc/orbit.conf.sample
@exec if [ ! -f %D/etc/orbit.conf ] ; then cp -p %D/%F %B/orbit.conf; fi
```

Данные строки являются упорядоченными. На этапе удаления файл с примером сравнивается с рабочим конфигурационным файлом. Полное совпадение означает отсутствие каких-либо изменений в рабочем файле со стороны пользователя, и следовательно этот файл может быть безопасно удалён. Так как файл с примером всё ещё должен существовать для сравнения, строка `@unexec` следует перед именем файла с примером конфигурации. На этапе установки, если рабочий файл конфигурации отсутствует, он копируется из файла с примером. Файл с примером обязательно должен быть установлен до операции копирования, поэтому строка `@exec` следует после имени файла с примером конфигурации.

Для получения дополнительного отладочного вывода на экран можно временно удалить параметр `-s` из команды cmp(1).

Для получения дополнительной инфорации по использованию `%D` и прочих маркеров подстановки обратитесь к странице Справочника pkg-create(8).

Если существует действительно стоящая причина не устанавливать рабочий файл конфигурации по умолчанию, уберите строку `@exec` из pkg-plist и добавьте сообщение, указывающее на то, что пользователь обязан скопировать и отредактировать этот файл перед тем, как программное обеспечение начнёт работать.

7.4. Динамический или статический список упаковки

Статический список упаковки - это список упаковки, который доступен в Коллекции Портов или как файл pkg-plist (с подстановкой переменных или без неё), или как встроенный в Makefile посредством `PLIST_FILES`, `PLIST_DIRS` и `PLIST_DIRSTRY`. Даже если содержимое является автоматически порождаемым при помощи инструмента или в результате выполнения цели в Makefile *до* включения в Коллекцию Портов коммиттером, то список всё ещё будет считаться статическим, поскольку его можно узнать без необходимости скачивания или компиляции дистрибутива.

Динамический список упаковки это список упаковки, который получается во время компиляции порта и строится на основе устанавливаемых файлов и каталогов. Узнать такой список невозможно до того, как исходный код портируемого приложения будет скачен и скомпилирован, или после запуска `make clean`.

Хотя использование динамических список упаковки не запрещено, сопровождающие должны использовать статические списки упаковки везде, где это возможно, поскольку это позволяет пользователям выполнять grep(1) по доступным портам для обнаружения, например, который порт устанавливает определенный файл. Динамические списки должны быть использованы в основном для сложных портов, для которых изменения в списке упаковки кардинальным образом основаны на необязательных возможностях порта (и, таким образом, делая сопровождение статических списков упаковки невозможным), или портов, которые изменяют список упаковки на основе версии используемого им программного обеспечения (например, порты, которые порождают документы при помощи Javadoc).

7.5. Автоматическое создание списка упаковки

Первым делом убедитесь, что ваш порт практически полностью завершён и осталось создать только pkg-plist. После этого вы можете запустить `make makeplist` для автоматического создания pkg-plist. Содержимое этого файла должно быть дважды перепроверено.

Пользовательские конфигурационные файлы должны быть удалены или быть установлены как filename.sample. Файл info/dir включать в список не нужно, но должны быть добавлены соответствующие строчки install-info, так, как это описано в разделе о файлах в формате info. Все библиотеки, устанавливаемые портом, должны быть перечислены так, как это описано в разделе о динамических библиотеках.

Глава 8. Файлы pkg-*

Есть несколько приёмов работы с файлами pkg-*, которые мы ещё не описали, но они иногда могут быть очень кстати.

8.1. pkg-message

Если вам нужно вывести сообщение для человека, устанавливающего приложение, то вы можете поместить сообщение в файл pkg-message. Эта возможность часто оказывается полезной для вывода дополнительных шагов установки, которые нужно предпринять после выполнения команды `pkg install`, или для вывода информации о лицензировании.

Если должны выводиться некоторые строки о knobs времени построения или предупреждения, используйте `ECHO_MSG`. Файл pkg-message только для послеустановочных шагов. Также следует иметь в виду различие между `ECHO_MSG` и `ECHO_CMD`. Первое предназначено для вывода на экран информационного текста, а второе для конвейера команд:

```
update-etc-shells:
    @${ECHO_MSG} "updating /etc/shells"
    @${CP} /etc/shells /etc/shells.bak
    @( ${GREP} -v ${PREFIX}/bin/bash /etc/shells.bak; \
        ${ECHO_CMD} ${PREFIX}/bin/bash) >/etc/shells
    @${RM} /etc/shells.bak
```

Файл pkg-message не нужно добавлять в pkg-plist.

8.2. pkg-install

Если при установке бинарного пакета по команде `pkg add` или `pkg install` вашему порту нужно выполнить какие-то команды, то вы можете это сделать с помощью скрипта pkg-install. Этот скрипт будет автоматически добавлен к пакету и будет дважды запускаться командой `pkg`: первый раз в виде `${SH} pkg-install ${PKGNAME} PRE-INSTALL`, а второй раз как `${SH} {PKGNAME} POST-INSTALL`. Для распознавания того, в каком режиме запущен скрипт, можно использовать параметр `$2`. Переменная окружения `PKG_PREFIX` будет принимать значение, соответствующее каталогу, в который устанавливается пакет.

Этот скрипт не запускается автоматически, если вы устанавливаете порт командой `make install`. Если же вам действительно необходимо его запустить, то запустите его явно из файла Makefile порта строкой вида `PKG_PREFIX=${PREFIX} ${SH} $ {PKGINSTALL}${PKGNAME} PRE-INSTALL`.

8.3. pkg-deinstall

Этот скрипт вызывается при удалении пакета.

Этот скрипт будет дважды запускаться командой `pkg delete`. Первый раз как `${SH} pkg-deinstall ${PKGNAME} DEINSTALL`, а второй раз как `${SH} pkg-deinstall ${PKGNAME} POST-DEINSTALL`.

8.4. Изменение имён файлов pkg-*

Все имена файлов pkg-* определяются с помощью переменных, так что вы можете изменить их, если это нужно, в вашем файле Makefile. Это особенно полезно, если вы используете одни и те же файлы pkg-* совместно между несколькими портами или пишете в один из вышеперечисленных файлов (в главе о "записи в каталоги, отличные от `WRKDIR`" объяснено, почему не рекомендуется осуществлять запись непосредственно в файлы pkg-*.

Вот список имён переменных и их значений по умолчанию. (Значение `PKGDIR` по умолчанию равно `${MASTERDIR}`.)

Переменная	Значение по умолчанию
`DESCR`	`${PKGDIR}/pkg-descr`
`PLIST`	`${PKGDIR}/pkg-plist`
`PKGINSTALL`	`${PKGDIR}/pkg-install`
`PKGDEINSTALL`	`${PKGDIR}/pkg-deinstall`
`PKGMESSAGE`	`${PKGDIR}/pkg-message`

Пожалуйста, изменяйте значения этих переменных, а не переопределяйте `PKG_ARGS`. Если вы измените значение переменных `PKG_ARGS`, то эти файлы при установке из порта будут установлены в каталог /var/db/pkg некорректно.

8.5. Использование `SUB_FILES` и `SUB_LIST`

Переменные `SUB_FILES` и `SUB_LIST` подходят для задания в файлах порта динамических значений, таких как `PREFIX` установки в pkg-message.

В переменной `SUB_FILES` указывается перечень файлов для автоматического изменения. Каждый *file* из перечня `SUB_FILES` обязан иметь соответствующий file.in, присутствующий в `FILESDIR`. Измененная версия будет создана в `WRKDIR`. Файлы, определенные в качестве значения `USE_RC_SUBR` (или устаревшего `USE_RCORDER`), автоматически добавляются в `SUB_FILES`. Для файлов pkg-message, pkg-install и pkg-deinstall устанавливается соответствующая переменная Makefile, указывающая на обработанную версию.

Переменная `SUB_LIST` содержит перечень пар `VAR=VALUE`. В каждом файле из `SUB_FILES` для каждой пары будет произведена замена `%%VAR%%` на `VALUE`. Некоторые общие пары определяются автоматически: `PREFIX`, `LOCALBASE`, `DATADIR`, `DOCSDIR`, `EXAMPLESDIR`, `WWWDIR` и `ETCDIR`. Любая строка, начинающаяся с `@comment`, будет удалена из конечного файла после подстановки переменной.

В следующем примере в pkg-message будет сделана замена `%%ARCH%%` на системную архитектуру:

```
SUB_FILES=  pkg-message
```

```
SUB_LIST=   ARCH=${ARCH}
```

Обратите внимание, что в этом примере в `FILESDIR` обязательно существование файла pkg-message.in.

Пример хорошего pkg-message.in:

```
Now it is time to configure this package.
Copy %%PREFIX%%/shared/examples/putsy/%%ARCH%%.conf into your home directory
as .putsy.conf and edit it.
```

Глава 9. Тестирование вашего порта

9.1. Запуск `make describe`

Некоторые утилиты FreeBSD для сопровождения портов, например, portupgrade(1), опираются на базу данных с именем /usr/ports/INDEX, в которой отслеживаются такие характеристики портов, как их зависимости. Файл INDEX создаётся при помощи ports/Makefile верхнего уровня по команде `make index`, спускающейся в подкаталог каждого порта и выполняющей в нём `make describe`. Таким образом, если выполнение `make describe` с каким-либо портом завершится неудачно, то никому не удастся создать INDEX, при этом много людей вскоре станут несчастны.

> Возможность генерировать этот файл очень важна вне зависимости от того, какие параметры присутствуют в make.conf, поэтому, пожалуйста, избегайте, таких вещей, как использование декларации `.error`, когда (к примеру) требования к зависимости не было удовлетворено. (Смотрите Избегайте использования конструкции `.error`.)

Если `make describe` выдаёт строчку, а не ошибку, то для вас это пройдёт безболезненно. Обратитесь к файлу bsd.port.mk, чтобы выяснить значение выдаваемых строк.

Заметьте также, что запуск последней версии `portlint` (как указано в следующем разделе) приведёт к автоматическому запуску команды `make describe`.

9.2. Portlint

Проверьте свою работу командой `portlint` перед тем, как её отослать или перенести в дерево портов. `portlint` предупреждает вас о многих распространённых ошибках, как функциональных, так и стилистических. Для нового (или скопированного внутри хранилища) порта самым подходящим является запуск `portlint -A`; для уже существующего порта достаточно будет запустить `portlint -C`.

Так как для обнаружения ошибок `portlint` использует эвристические методы, то им могут выдаваться и ошибочные предупреждения. Кроме того, время от времени нечто, отмечаемое как некорректность, из-за ограничений механизма создания портов не может быть сделано никак иначе. Если вы сомневаетесь, то лучше всего спросить в Список рассылки, посвящённый Портам FreeBSD.

9.3. Port Tools

Программа ports-mgmt/porttools входит в состав Коллекции Портов.

`port` является сценарием переднего плана, который может упростить вам задачу тестирования. Если вы хотите проверить новый порт или обновить существующий, то вы можете использовать `port test` для проверки вашего порта, включая проверку `portlint`. Эта команда также находит и отображает любые файлы, которые невключенные в pkg-plist. Смотрите следующий пример:

```
# port test /usr/ports/net/csup
```

9.4. PREFIX и DESTDIR

Переменная PREFIX определяет, куда будет установлен порт. По умолчанию это /usr/local, но может меняться пользователем на собственный путь, такой как /opt. В вашем порту значение этой переменной должно учитываться.

Если пользователь установил переменную DESTDIR, то она определяет полное альтернативное окружение, обычно, это jail или установленная система, смонтированная в месте, отличном от /. На самом деле порт устанавливается в DESTDIR/PREFIX и регистрируется в базе данных пакетов в DESTDIR/var/db/pkg. Поскольку управление DESTDIR производится автоматически инфраструктурой портов с помощью chroot(8), вам не нужны никакие изменения или проявление особой осторожности при написании портов, совместимых с DESTDIR.

Значение переменной PREFIX будет установлено в LOCALBASE (по умолчанию /usr/local). Если задана переменная USE_LINUX_PREFIX, то PREFIX примет значение LINUXBASE (по умолчанию /compat/linux).

Избегание явно прописываемых путей /usr/local в исходном коде сделает порт гораздо более гибким и способным удовлетворить потребности других серверов. Часто этого можно добиться простой заменой строк /usr/local в различных файлах Makefile внутри порта на ${PREFIX}. Эта переменная автоматически передаётся далее на каждом этапе построения и установки.

Проверьте, что ваше приложение не устанавливает чего-либо в каталог /usr/local вместо PREFIX. Наличие явно указанных путей можно быстро проверить следующим образом:

```
# make clean; make package PREFIX=/var/tmp/`make -V PORTNAME`
```

Если что-то было установлено за пределами PREFIX, то процесс создания пакета сообщит об отсутствии файлов.

Это также стоит проверить с использованием поддержки каталога сборки (смотрите Staging):

```
# make stage && make check-orphans && make package
```

Эти проверки не найдут явно указанных путей внутри файлов порта и не проверят корректность использования LOCALBASE в качестве ссылки на файлы из других портов. Порт, временно установленный в /var/tmp/make -V PORTNAME, следует проверять на работоспособность, чтобы убедиться в отсутствии проблем с путями.

Переменная PREFIX не должна задаваться явно в файле Makefile порта. Пользователи при установке порта могут задать в PREFIX свое собственное место, и порт должен учитывать это

значение.

Обратитесь к программам/файлам из других портов с переменными, перечисленными выше, без указания явных маршрутов. Например, если ваш порт требует, чтобы макрос `PAGER` являлся полным путем утилиты `less`, не используйте строковый путь /usr/local/bin/less. Вместо этого используйте `${LOCALBASE}`:

```
-DPAGER=\"${LOCALBASE}/bin/less\"
```

Путь с использованием `LOCALBASE` имеет больше шансов оставаться работоспособным, если системный администратор переместил всё дерево /usr/local куда-то в другое место.

9.5. Tinderbox

Если вы алчный контрибутор портов, то вы можете захотеть взглянуть на Tinderbox. Это мощная система построения и тестирования портов. Tinderbox можно установить, используя порт ports-mgmt/tinderbox. Обязательно прочитайте поставляемую документацию, поскольку конфигурация не является тривиальной.

Для получения подробностей посетите вебсайт Tinderbox.

9.6. Poudriere

Если вы контрибутор портов, подумайте об установке poudriere. Это мощная система для построения и тестирования портов. Poudriere можно установить из ports-mgmt/poudriere.

Для получения подробной информации посетите вебсайт Poudriere.

Глава 10. Обновление отдельного порта

Если вы заметите, что ваш порт устарел по сравнению с последней авторской версией, первым делом вы должны получить самую последнюю версия порта. Вы можете найти их в каталоге ports/ports-current на зеркальных FTP-серверах FreeBSD. Однако если вы работаете с достаточно большим количеством портов, наверное, будет проще использовать Subversion или portsnap(8) для поддержания всей коллекции портов в актуальном состоянии, как это описано в Руководстве. К тому же это даст возможность отслеживать все зависимости портов.

На следующем шаге необходимо выяснить, нет ожидает ли уже это обновление своей очереди. Для этого у вас есть две возможности. Существует интерфейс к базе данных сообщений о проблемах FreeBSD (PR) (известной также как `GNATS`) с поисковыми возможностями. Выберите из выпадающего списка `ports` и введите название порта.

Однако иногда люди забывают поместить название порта в поле Synopsis в точном виде. В таком случае вы можете воспользоваться Системой мониторинга портов FreeBSD (которая известна также как `portsmon`). В рамках этой системы делается попытка классифицировать PR, касающиеся портов, по имени порта. Для поиска PR, относящихся к определённому порту, используйте механизм Просмотра по одному порту.

Если таких отложенных PR не существует, то на следующем этапе следует послать сообщение электронной почты человеку, поддерживающему порт, который выдаётся по команде `make maintainer`. Этот человек может уже работать над обновлением, или иметь причину не обновлять порт прямо сейчас (например, из-за проблем со стабильностью функционирования новой версии); вам нет нужды дублировать их работу. Заметьте, что неподдерживаемые порты перечисляются с адресом сопровождающего `ports@FreeBSD.org`, который является всего лишь адресом общего списка рассылки, так что отправка туда сообщений, скорее всего, в данном случае не поможет.

Если сопровождающий просит вас выполнить обновление, либо сопровождающий отсутствует, то у вас появляется шанс помочь FreeBSD, приготовив обновление самим! Пожалуйста, делайте это с использованием команды diff(1) в основной системе.

Чтобы создать подходящий `diff` для одного патча, скопируйте файл, который нужно пропатчить, в *something.orig*, сохраните ваши изменения в *something*, а затем создайте ваше патч:

```
% diff -u something.orig something > something.diff
```

В противном случае, вам следует воспользоваться методом `svn diff` (Использование Subversion для создания патчей), либо скопировать содержимое порта в отдельный каталог и применить результат рекурсивной команды diff(1) между новым и старым каталогами порта (например, если каталог с модифицированным портом называется superedit, а оригинальный, совпадающий с находящимся в нашем дереве портов, superedit.bak, то сохраните результат выполнения команды `diff -ruN superedit.bak superedit`). Подойдёт как унифицированный, так и контекстный дифф, однако коммиттеры портов обычно

предпочитают унифицированный формат. Отметьте использование опции `-N`-это одобряемый способ заставить diff корректно работать в случае добавления новых файлов или удаления старых. Перед тем, как посылать нам diff-файл, пожалуйста, проверьте его, чтобы убедиться в значимости всех внесённых изменений. (В частности, убедитесь, что вы очистили рабочие каталоги командой `make clean`).

Для упрощения повторяющихся операций с файлами заплаток вы можете воспользоваться скриптом /usr/ports/Tools/scripts/patchtool.py. Перед тем, как его запускать, пожалуйста, прочтите /usr/ports/Tools/scripts/README.patchtool.

Если порт никем не поддерживается, а вы активно его используете, пожалуйста, подумайте над тем, чтобы добровольно стать его сопровождающим. Во FreeBSD имеется более 4000 портов без поддержки, и это как раз та область, где всегда нужны добровольцы. (Детальное описание обязанностей сопровождающего можно найти в разделе Руководства Разработчика.)

Лучше всего послать нам diff-файл, включив его в посылку по команде send-pr(1) (категория `ports`). Если вы сопровождаете порт, обязательно поместите текст `[maintainer update]` в начале строки описания и задайте в поле "Class" вашего PR строчку `maintainer-update`. В противном случае в поле "Class" вашего PR должно быть указано `change-request`. Будьте добры, в сообщении отметьте все добавленные или удалённые файлы, так как они будут непосредственно указаны svn(1) при выполнении операции коммита. Если diff-файл имеет размер, превышающий 20КБ, сожмите его и обработайте утилитой uuencode; в противном случае просто включите его как есть в PR.

Прежде чем пользоваться send-pr(1) просмотрите раздел о Написании сообщений о проблемах в статье о Сообщениях об ошибках. Он содержит гораздо больше информации о том, как писать полезные сообщения о проблемах.

> Если обновление вызвано соображениями информационной безопасности или наличием серьёзных ошибок в имеющемся порте, пожалуйста, оповестите Группа Менеджеров Дерева Портов FreeBSD <portmgr@FreeBSD.org> о необходимости немедленного перепостроения и повторного распространения пакета данного порта. В противном случае ничего не подозревающие пользователи `pkg` будут продолжать устанавливать старую версию по команде `pkg install` в течение ещё нескольких недель.

> Повторяем еще раз - для посылки обновлений существующих портов используйте утилиту diff(1), а не shar(1)! Это поможет понять коммиттерам портов, что именно было изменено.

Теперь, когда вы проделали всё это, прочитайте о том, как поддерживать актуальное состояние, в Актуализация.

10.1. Использование Subversion для создания патчей

По возможности присылайте исправления в формате svn(1) diff. В таком виде их проще использовать по сравнению с разницей между "старым и новым" каталогами. Так проще увидеть изменения и обновить их в случае, если что-нибудь изменилось в Коллекции Портов с тех пор, как вы начали работу, либо если коммиттер просит что-то исправить.

```
% cd ~/my_wrkdir ①
% svn co https://svn0.us-west.FreeBSD.org/ports/head/dns/pdnsd ②
% cd ~/my_wrkdir/pdnsd
```

① Это может быть где угодно; место, в котором производится построение портов, не привязано к /usr/ports/.

② svn0.us-west.FreeBSD.org - это общедоступный сервер Subversion. Выберите ближайшее зеркало и проверьте сертификат зеркалирующего сервера на наличие в перечне зеркалирующих сайтов Subversion.

Находясь в рабочем каталоге, вносите любые изменения, которые обычно делают для порта. При добавлении или удалении файла используйте `svn` для отслеживания этих изменений:

```
% svn add new_file
% svn remove deleted_file
```

Убедитесь, что вы проверяете порт в соответствии с рекомендуемым порядком проверки, описанным в Тестирование порта и Проверка вашего порта утилитой `portlint`.

```
% svn status
% svn update ①
```

① Эта команда попытается выполнить слияние различий между вашим патчем и текущей версией репозитория; внимательно проверьте полученный вывод. Буква перед названием каждого файла означает тип изменения, сделанного с этим файлом. Для получения полного списка смотрите Префиксы файлов для Subversion Update.

Таблица 48. Префиксы файлов для Subversion Update

U	Файл обновлен без проблем.
G	Файл обновлен без проблем (вы увидите это только при работе с удаленным репозиторием).
M	Файл с локальными изменениями, слияние выполнено без конфликтов.
C	Файл с локальными изменениями, слияние выполнено с конфликтами.

Если в результате выполнения `svn update` отображается `C`, то это означает, что что-то изменилось в репозитории Subversion и svn(1) не смогла выполнить слияние локальных изменений с полученными из репозитория. В любом случае никогда не помешает просмотреть изменения, поскольку svn(1) ничего не знает о том, каким должен быть порт, поэтому эта команда может (и, вероятно, будет) делать слияние тех изменений, которые не имеют смысла.

Последним шагом является создание унифицированного diff(1) для полученных изменений:

```
% svn diff > ../`basename ${PWD}`.diff
```

Информация о любых удаляемых файлов должна быть явным образом указана в PR, поскольку необходимость в удалении файла для коммиттера может быть неочевидна.

Присылайте свои патчи в соответствии с руководством, описанном в Обновление отдельного порта.

10.2. Файлы UPDATING и MOVED

Если при обновлении порта требуются специальные шаги, такие как изменение файлов конфигурации или запуск специальной программы, то вам следует это задокументировать в файле /usr/ports/UPDATING. Формат записи в этом файле приводится ниже:

```
YYYYMMDD:
  AFFECTS: users of portcategory/portname
  AUTHOR: Your name <Your email address>

  Special instructions
```

Если вы включаете точные инструкции portmaster или portupgrade, пожалуйста, убедитесь в правильном экранировании символов внутри командной оболочки.

Файл /usr/ports/MOVED содержит записи об удалённых или перемещённых портах. Каждая строка в этом файле состоит из полей: название порта, место, куда он был перемещён, дата и причина перемещения. Если порт был удалён, то поле, указывающее новое место, может оставаться незаполненным. Поля должны разделяться символом `|` (pipe), как это показано ниже:

```
old name|new name (blank for deleted)|date of move|reason
```

Дату следует вводить в формате `YYYY-MM-DD`. Новые записи следует добавлять в конец файла в хронологическом порядке.

Если порт был перемещён, но в дальнейшем восстановлен на прежнем месте, удалите в

этом файле строку, содержащую информацию о перемещении.

Полученные изменения можно проверить командой `Tools/scripts/MOVEDlint.awk`.

Глава 11. Безопасность портов

11.1. Почему безопасность так важна

Ошибки в программном обеспечении появляются случайно. Возможно, самые опасные из них те, что создают уязвимости безопасности. С технической точки зрения подобные уязвимости должны быть закрыты путем исправления вызывающих их ошибок. Тем не менее, политики обработки несущественных ошибок и уязвимостей очень различаются.

Обычная небольшая ошибка затрагивает только тех пользователей, которые задействуют некоторые комбинации настроек, активирующие эту ошибку. Разработчик в конечном счете выпустит патч, а зачтем новую версию программного обеспечения, свободного от ошибки, но большинство пользователей не посчитают нужным сразу же произвести обновление, поскольку эта ошибка никогда у них не проявлялась. Критическая ошибка, которая может приводить к потере данных, представляет серьезную проблему. Тем не менее, предусмотрительные пользователи знают, что большинство возможных происшествий, и среди них программные ошибки, скорее всего приводят к потере данных, поэтому они выполняют резервное копирование важных данных; дополнительно, критическая ошибка будет обнаружена очень скоро.

С уязвимостью безопасности всё иначе. Во-первых, она может сохраняться необнаруженной целые годы, потому что чаще всего не вызывает ошибок в работе. Во-вторых, компания злоумышленников может использовать ее для получения неавторизованного доступа к уязвимой системе, уничтожить или подменить важные данные; в худшем случае пользователь даже не заметит нанесенный урон. В-третьих, взлом уязвимой системы часто упрощает задачу проникновения атакующих в другие системы, которые не могут быть скомпрометированы иначе. Таким образом, устранение уязвимости как таковой недостаточно: следует разослать всем заинтересованным уведомления в наиболее понятной и исчерпывающей форме, что позволит оценить риск и предпринять подходящие меры.

11.2. Исправление уязвимостей безопасности

Что касается портов и пакетов, уязвимость безопасности изначально может появиться в исходном дистрибутиве или файлах порта. В первом случае, разработчик исходного программного обеспечения скорее всего сразу же выпустит патч или новую версию, и вам лишь понадобится сразу обновить порт в соответствии с исправлением автора. Если исправление по какой-то причине задерживается, вам следует либо пометить порт как `FORBIDDEN`, либо добавить в порт ваш собственный патч. В случае уязвимости порта просто исправьте этот порт как можно скорее. В любом случае нужно следовать стандартной процедуре отправки вашего изменения, если вы не обладаете правами на коммит изменения непосредственно в дерево портов.

Быть коммиттером портов недостаточно для коммита произвольного порта. Помните, что обычно у портов есть сопровождающие, мнение которых вы должны учитывать.

Пожалуйста, убедитесь, что ревизия порта после закрытия уязвимости увеличена. Вот как пользователи, обновляющие установленные пакеты на постоянной основе, увидят, что им нужно запустить обновление. Кроме того, новый пакет будет собран и распространен через FTP и WWW зеркала, замещая уязвимый. Если в процессе исправления уязвимости не было изменено значение `PORTVERSION`, то должно быть увеличено значение `PORTREVISION`. Вам следует увеличить значение `PORTREVISION` после добавления в порт файла с патчем, но не когда вы обновили порт до последней версии программного обеспечения, попутно затронув при этом `PORTVERSION`. За дальнейшей информацией обращайтесь к соответствующему разделу.

11.3. Обеспечение сообщества информацией

11.3.1. База данных VuXML

Очень важным и первостепенным шагом при действии как можно раньше после раскрытия уязвимости является уведомление сообщества пользователей порта об опасности. Такие уведомления служат двум целям. Во-первых, в случае действительно серьезной угрозы, будет посоветовано применить мгновенное воздействие. Например, остановить затрагиваемый сетевой сервис или даже удалить порт целиком, пока уязвимость не будет устранена. Во-вторых, масса пользователей имеет тенденцию обновлять установленные пакеты только от случая к случаю. Из уведомления они узнают, что *должны* обновить пакет без промедления сразу же после появления исправленной версии.

Учитывая огромное число портов в дереве, невозможно по каждому случаю выпускать бюллетень безопасности без создания флуда и потери внимания сообщества к моменту появления действительно серьезных причин. Поэтому уязвимости безопасности, обнаруженные в портах, записываются в базу данных FreeBSD VuXML. Члены Команды Офицеров Безопасности также отслеживают её на предмет появления вопросов, требующих их вмешательства.

Если вы обладаете правами коммиттера, вы можете сам обновить базу данных VuXML. Так вы поможете Команде Офицеров Безопасности и своевременно пошлете ценную информацию сообществу. Тем не менее, если вы не являетесь коммиттером или верите, что нашли исключительно серьезную уязвимость, то не задумываясь свяжитесь с Командой Офицеров Безопасности напрямую как это описано на странице информационной безопасности FreeBSD.

База данных VuXML является документом XML. Его исходный файл vuln.xml содержится прямо внутри порта security/vuxml. Следовательно, полное имя пути к файлу будет PORTSDIR/security/vuxml/vuln.xml. Каждый раз, при обнаружении вами в порте уязвимости безопасности добавьте об этом запись в этот файл. Пока вы не знакомы с VuXML, лучшее, что вы можете сделать, это найти существующую запись, подпадающую под ваш случай, затем скопировать ее и использовать в качестве шаблона.

11.3.2. Короткое вступление в VuXML

В совокупности XML является очень сложным форматом, и его описание выходит далеко за

рамки этой книги. Тем не менее, для достижения основного понимания структуры записи VuXML вам понадобится всего лишь понять теги. Имена тегов XML обрамляются в угловые скобки. Каждый открывающий <tag> должен иметь совпадающий закрывающий </tag>. Теги могут быть вложенными. При вложенности внутренние теги должны быть закрыты до закрытия внешних. Существует иерархия тегов, т.е. более сложные правила вкладывания тегов. Это похоже на HTML. Основное отличие в расширяемости XML, т.е. в определении собственных тегов. Из-за своей характерной структуры XML придает форму разрозненным данным. В частности, XML подходит для разметки описаний уязвимостей безопасности.

Теперь рассмотрим настоящую запись VuXML:

```
<vuln vid="f4bc80f4-da62-11d8-90ea-0004ac98a7b9"> ①
  <topic>Several vulnerabilities found in Foo</topic> ②
  <affects>
    <package>
      <name>foo</name> ③
      <name>foo-devel</name>
      <name>ja-foo</name>
      <range><ge>1.6</ge><lt>1.9</lt></range> ④
      <range><ge>2.*</ge><lt>2.4_1</lt></range>
      <range><eq>3.0b1</eq></range>
    </package>
    <package>
      <name>openfoo</name> ⑤
      <range><lt>1.10_7</lt></range> ⑥
      <range><ge>1.2,1</ge><lt>1.3_1,1</lt></range>
    </package>
  </affects>
  <description>
    <body xmlns="http://www.w3.org/1999/xhtml">
      <p>J. Random Hacker reports:</p> ⑦
      <blockquote
        cite="http://j.r.hacker.com/advisories/1">
        <p>Several issues in the Foo software may be exploited
          via carefully crafted QUUX requests.  These requests will
          permit the injection of Bar code, mumble theft, and the
          readability of the Foo administrator account.</p>
      </blockquote>
    </body>
  </description>
  <references> ⑧
    <freebsdsa>SA-10:75.foo</freebsdsa> ⑨
    <freebsdpr>ports/987654</freebsdpr> ⑩
    <cvename>CAN-2010-0201</cvename> ⑪
    <cvename>CAN-2010-0466</cvename>
    <bid>96298</bid> ⑫
    <certsa>CA-2010-99</certsa> ⑬
    <certvu>740169</certvu> ⑭
    <uscertsa>SA10-99A</uscertsa> ⑮
    <uscertta>SA10-99A</uscertta> ⑯
```

```
    <mlist
msgid="201075606@hacker.com">http://marc.theaimsgroup.com/?l=bugtraq&amp;m=20388660782
5605</mlist> ⑰
    <url>http://j.r.hacker.com/advisories/1</url> ⑱
  </references>
  <dates>
    <discovery>2010-05-25</discovery> ⑲
    <entry>2010-07-13</entry> ⑳
    <modified>2010-09-17</modified>
  </dates>
</vuln>
```

Имена тегов должны быть самодокументируемыми, чтобы мы сфокусировались только на полях, нужных нам для заполнения:

① Это тег верхнего уровня записи VuXML. У него есть обязательный атрибут `vid`, указывающий на универсальный уникальный идентификатор (UUID) для этой записи (в кавычках). Вы должны формировать UUID для каждой новой записи VuXML (и не забудьте заменить ее для шаблона UUID, если вы не пишете запись с нуля). Для получения VuXML UUID вы можете использовать uuidgen(1).

② Однострочное описание найденной проблемы.

③ Здесь перечислены имена затронутых пакетов. Может быть дано несколько имен, поскольку некоторые пакеты могут быть основаны на одном главном порте или программном продукте. Сюда можно включить стабильную ветвь и ветвь разработки, локализованные версии и подчиненные порты, зависящие от различного выбора важных вариантов конфигурации, указанных на этапе построения.

④ Здесь указаны затронутые версии пакета(-ов) как один или более диапазонов с использованием комбинации элементов `<lt>`, `<le>`, `<eq>`, `<ge>`, и `<gt>`. Диапазоны внесённых версий не должны пересекаться.В спецификации диапазонов `*` (звёздочка) означает наименьший номер версии. В частности, `2.*` меньше, чем `2.a`. Поэтому звездочка может быть использована в диапазоне для совпадения со всеми возможными `alpha`, `beta` и `RC` версиями. Как вариант, `<ge>2.</ge><lt>3.</lt>` выборочно совпадет с версией `2.x`, а `<ge>2.0</ge><lt>3.0</lt>` - нет, поскольку последнее не включает `2.r3` и совпадает с `3.b`.Пример выше указывает, что к затронутым относятся версии с `1.6` до `1.9` включительно, версии `2.x` до `2.4_1` и версия `3.0b1`.

⑤ Некоторые связанные группы пакетов (в конечном счете, порты) могут быть указаны в разделе `<affected>`. Это можно использовать, если некоторые программные продукты (скажем, FooBar, FreeBar and OpenBar) являются производными от общей кодовой базы и всё еще совместно используют её ошибки и уязвимости. Имейте в виду отличие от перечисления множественных имён в одном разделе <package>.

⑥ Диапазоны версий должны учитывать `PORTEPOCH` и `PORTREVISION`, если это применимо. Пожалуйста, помните, что в соответствии с правилами сравнения строк версия с ненулевым значением `PORTEPOCH` выше, чем любая версия без `PORTEPOCH`, например, `3.0,1` выше, чем `3.1` или даже `8.9`.

⑦ Сводная информация о проблеме. В этом поле используется XHTML. По крайней мере, должны быть обрамляющие `<p>` и `</p>`. Может быть использована более сложная

разметка, но только в целях аккуратности и ясности: без эстетства, пожалуйста.

⑧ Этот раздел содержит ссылки на имеющие отношение документы. Приветствуется как можно большее количество ссылок.

⑨ Это бюллетень безопасности FreeBSD.

⑩ Это сообщение об ошибке FreeBSD.

⑪ Идентификатор MITRE CVE.

⑫ Это SecurityFocus Bug ID.

⑬ Бюллетень безопасности US-CERT.

⑭ Примечание к уязвимости US-CERT.

⑮ Уведомление системы Cyber Security Alert US-CERT.

⑯ Уведомление системы Technical Cyber Security Alert US-CERT.

⑰ URL к архивному сообщению в списке рассылки. Атрибут `msgid` является необязательным и может указывать на message ID сообщения.

⑱ Основной URL. Должен быть использован в случае, если не подходит ни одна из категорий источника.

⑲ Дата последнего изменения любой информации данной записи (*YYYY-MM-DD*). Новые записи не должны включать это поле. Поле должно быть добавлено после редактирования существующей записи.

11.3.3. Тестирование ваших изменений в базе данных VuXML

Предположим, что вы только что написали или заполнили запись об уязвимости в пакете `clamav`, которая была исправлена в версии `0.65_7`.

Прежде всего, вам нужно *установить* последние версии портов ports-mgmt/portaudit, ports-mgmt/portaudit-db и security/vuxml.

> Для запуска `packaudit` вы должны обладать правами на запись в DATABASEDIR; как правило, это /var/db/portaudit.
>
> Для использования другого каталога присвойте переменной окружения DATABASEDIR другой путь.
>
> Если вы работаете в каталоге, отличном от ${PORTSDIR}/security/vuxml, присвойте переменной окружения VUXMLDIR путь к каталогу, в котором находится vuln.xml.

Во-первых, проверьте, нет ли уже записи об этой уязвимости. Если такая запись есть, она совпадёт с предыдущей версией пакета `0.65_6`:

```
% packaudit
% portaudit clamav-0.65_6
```

Если ничего не найдено, значит вы получили зеленый свет для добавления новой записи для этой уязвимости.

```
% cd ${PORTSDIR}/security/vuxml
% make newentry
```

Когда вы закончите, проверьте синтаксис и форматирование.

```
% make validate
```

Вам понадобится установить по крайней мере один из следующих пакетов: textproc/libxml2, textproc/jade.

Теперь выполните перепостроение базы данных `portaudit` из файла VuXML:

```
% packaudit
```

Чтобы убедиться, что раздел `<affected>` в вашей записи совпадает с правильными пакетами, выполните следующую команду:

```
% portaudit -f /usr/ports/INDEX -r uuid
```

Для лучшего понимания синтаксиса этой команды обращайтесь к portaudit(1).

Убедитесь, что ваша запись не производит ложных совпадений в выводе.

Теперь проверьте, совпадает ли ваша запись с нужными версиями пакета:

```
% portaudit clamav-0.65_6 clamav-0.65_7
Affected package: clamav-0.65_6 (matched by clamav<0.65_7)
Type of problem: clamav remote denial-of-service.
Reference: <http://www.freebsd.org/ports/portaudit/74a9541d-5d6c-11d8-80e3-
0020ed76ef5a.html>

1 problem(s) found.
```

Первая версия должна совпасть, а последняя нет.

В заключение проверьте, что веб-страница, сформированная из базы данных VuXML, выглядит как положено:

```
% mkdir -p ~/public_html/portaudit
```

```
% packaudit
% lynx ~/public_html/portaudit/74a9541d-5d6c-11d8-80e3-0020ed76ef5a.html
```

Глава 12. Что делать нужно, и что делать нельзя

12.1. Введение

Вот список часто встречающихся действий, которые нужно и которые нельзя делать во время процесса портирования. Проверьте порт по этому списку, и также проверьте порты в базе сообщений PR, которые присланы другими людьми. Присылайте любые комментарии о портах, которые вы проверили, так, как это описано в статье о Сообщениях об ошибках и общих замечаниях. Проверка портов в базе сообщений PR позволит нам быстрее коммиттить их и удостовериться, что вы знаете, что делаете.

12.2. `WRKDIR`

Не пишите ничего в файлы вне каталога `WRKDIR`. Каталог `WRKDIR` является единственным местом, которое гарантированно будет доступно для записи во время построения порта (обратитесь к главе об установке портов с CDROM за примером построения портов из дерева, доступного только для чтения). Если вам нужно изменить какой-либо из файлов pkg-*, сделайте это, переопределив переменную, но не перезаписывая их.

12.3. `WRKDIRPREFIX`

Добейтесь того, чтобы ваш порт принимал во внимание значение переменной `WRKDIRPREFIX`. Большинство портов об этом не заботятся. В частности, если вы обращаетесь к каталогу `WRKDIR` другого порта, заметьте, что его правильным местоположением является WRKDIRPREFIXPORTSDIR/subdir/name/work, а не PORTSDIR/subdir/work или .CURDIR/../../subdir/name/work мли что-то подобное.

Кроме того, если вы сами задаете `WRKDIR`, то должны поставить перед ним знак `${WRKDIRPREFIX}${.CURDIR}`.

12.4. Различение операционных систем и версий ОС

Вы можете встретиться с кодом, который требует модификаций или условной компиляции в зависимости от того, с какой версией FreeBSD Unix он работает. Предпочтительным способом отделения кода для версий FreeBSD является использование макросов __FreeBSD_version и __FreeBSD__, определённых в sys/param.h. Если этот файл не подключен, добавьте код

```
#include <sys/param.h>
```

в нужном месте файла .c.

__FreeBSD__ определён во всех версиях FreeBSD в качестве старшего номера версии системы. Например, в FreeBSD 9.x __FreeBSD__ определён со значением 9.

```
#if __FreeBSD__ >= 9
#  if __FreeBSD_version >= 901000
     /* здесь особый код для версий 9.1+ */
#  endif
#endif
```

12.5. Написание чего-либо после bsd.port.mk

Не пишите ничего после строки `.include <bsd.port.mk>`. Этой строки можно избежать, включив в где-то в середину вашего файла Makefile файл bsd.port.pre.mk, и файл bsd.port.post.mk в конец.

Вам нужно включить либо пару файлов bsd.port.pre.mk/bsd.port.post.mk, либо только bsd.port.mk; не используйте оба этих метода одновременно.

В файле bsd.port.pre.mk определяются лишь несколько переменных, которые могут быть использованы в тестах из файла Makefile, в файле bsd.port.post.mk заданы остальные.

Вот некоторые важные переменные, определенные в файле bsd.port.pre.mk (это не полный список, для выяснения полного списка прочтите, пожалуйста, сам файл bsd.port.mk).

Переменная	Описание
`ARCH`	Архитектура машины в виде, получаемом по команде `uname -m` (например, `i386`)
`OPSYS`	Тип операционной системы, получаемый по команде `uname -s` (например, `FreeBSD`)
`OSREL`	Версия релиза операционной системы (например, `2.1.5` или `2.2.7`)
`OSVERSION`	Версия операционной системы в виде числа, та же, что и `__FreeBSD_version`.
`LOCALBASE`	Корень дерева "local" (например, `/usr/local`)
`PREFIX`	Куда, собственно, устанавливается порт (обратитесь к подробной информации о `PREFIX`).

Если вы задаете переменную `MASTERDIR`, делайте это до подключения bsd.port.pre.mk.

Вот несколько примеров того, что вы можете написать после bsd.port.pre.mk:

```
# no need to compile lang/perl5 if perl5 is already in system
```

```
.if ${OSVERSION} > 300003
BROKEN= perl is in system
.endif
```

Вы не забываете об использовании табуляции вместо пробелов после `BROKEN=`:-).

12.6. Использование выражения `exec` в сценариях обёртках

Если порт устанавливает сценарий на языке shell, который служит для запуска другой программы, и если запуск этой программы является последним действием сценария, убедитесь, что запуск программы производится с использованием выражения `exec`, например:

```
#!/bin/sh
exec %%LOCALBASE%%/bin/java -jar %%DATADIR%%/foo.jar "$@"
```

Выражение `exec` заменяет процесс сценария на указанную программу. Если `exec` опущен, то процесс сценария во время работы программы остается в памяти, бесполезно потребляя системные ресурсы.

12.7. Поступайте разумно

Файл Makefile должен выполнять действия просто и небеспричинно. Если вы можете сделать что-то на несколько строк короче или более читабельно, сделайте это. В качестве примеров можно привести использование конструкций `.if` утилиты make вместо соответствующей конструкции `if` командного процессора, ненужность переопределения цели `do-extract` при возможности переопределения `EXTRACT*` и использование `GNU_CONFIGURE` вместо `CONFIGURE_ARGS += --prefix=${PREFIX}`.

Если вы обнаружите, что для выполнения чего-то приходится писать много нового кода, то, пожалуйста, просмотрите файл bsd.port.mk на предмет того, не содержит ли он решение именно вашей проблемы. Хотя его трудно читать, имеется много проблем, выглядящих сложными, для которых файл bsd.port.mk уже содержит быстрое решение.

12.8. Работа как с `CC`, так и `CXX`

Порт должен принимать во внимание как переменную `CC`, так и `CXX`. Под этим мы подразумеваем, что порт ни в коем случае не должен устанавливать значения этих переменных, переопределяя имеющиеся значения; вместо этого можно добавлять нужные значения к уже имеющимся. Это связано с тем, что параметры построения, относящиеся ко всем портам, могут быть заданы глобально.

Если порты не учитывают значения этих переменных, добавьте строку `NO_PACKAGE=ignores either cc or cxx` в файл Makefile.

Далее следует пример файла Makefile, использующего как переменную `CC`, так и `CXX`. Обратите внимание на использование символов `?=`:

```
CC?= gcc
```

```
CXX?= g++
```

Вот пример, в котором не принимаются во внимание ни `CC`, ни `CXX`:

```
CC= gcc
```

```
CXX= g++
```

В системах FreeBSD обе переменные `CC` и `CXX` могут быть определены в файле /etc/make.conf. В первом примере задаётся значение, если оно ранее не было определено в /etc/make.conf, что сохраняет любые определения, данные на уровне системы в целом. Второй пример переопределяет всё, что было задано ранее.

12.9. Использование `CFLAGS`

Порт должен учитывать переменную `CFLAGS`. Под этим мы подразумеваем, что порт ни в коем случае не должен устанавливать значения этой переменной, переопределяя имеющиеся значения; вместо этого можно добавлять нужные значения к уже имеющимся. Это связано с тем, что параметры построения, относящиеся ко всем портам, могут быть заданы глобально.

Если порты не учитывают значения этой переменной, добавьте строку `NO_PACKAGE=ignores cflags` в файл Makefile.

Далее следует пример файла Makefile, использующего переменную `CFLAGS`. Обратите внимание на использование символов `+=`:

```
CFLAGS+= -Wall -Werror
```

А вот пример, в котором не учитывается значение переменной `CFLAGS`:

```
CFLAGS= -Wall -Werror
```

В системе FreeBSD переменная `CFLAGS` определена в файле /etc/make.conf. В первом примере к переменной `CFLAGS` добавляются дополнительные флаги, при этом сохраняются все определения, данные ранее на уровне системы. Во втором примере всё, что было задано ранее, игнорируется.

Из сторонних файлов Makefile следует удалить флаги оптимизации. Общесистемные флаги оптимизации находятся в системной переменной `CFLAGS`. Пример из немодифицированного Makefile:

```
CFLAGS= -O3 -funroll-loops -DHAVE_SOUND
```

При использовании системных флагов оптимизации Makefile станет похожим на следующий пример:

```
CFLAGS+= -DHAVE_SOUND
```

12.10. Библиотеки потоков

Во FreeBSD библиотека потоков обязана быть скомпонована с исполняемыми файлами с использованием специального флага `-pthread`. Если порт настаивает на прямой компоновке с `-lpthread`, создайте патч для использования `-pthread`.

> Если построение порта заканчивается ошибкой `unrecognized option '-pthread'`, то может быть желательно использование `cc` в качестве компоновщика через установку `CONFIGURE_ENV` в `LD=$Cheng Cui <cc@FreeBSD.org>`. Параметр `-pthread` напрямую командой `ld` не поддерживается.

12.11. Пожелания

Посылайте подходящие изменения/патчи автору/сопровождающему для включения в следующий релиз. Это только сделает вашу работу гораздо легче при выходе следующего релиза.

12.12. README.html

README.html не является частью порта и генерируется при помощи `make readme`. Не включайте этот файл в патчи или коммиты.

> Если не удается выполнить `make readme`, убедитесь, что значение по умолчанию `ECHO_MSG` не изменено внутри порта.

12.13. Пометка неустанавливаемого порта как `BROKEN`, `FORBIDDEN` или `IGNORE`

В некоторых случаях пользователи не должны допускаться к установке порта. Для того, чтобы сообщить пользователю, что порт не следует устанавливать, имеется несколько `make`-переменных, которые могут быть использованы в файле Makefile порта. Значения следующих `make`-переменных будут причиной, возвращаемой пользователям, по которой

порт отказывает в установке. Пожалуйста, используйте корректные `make`-переменные, так как каждая переменная make передает абсолютно различный смысл как для пользователей, так и для автоматизированных систем, которые полагаются на файлы Makefile, таких как кластер построения портов, FreshPorts и portsmon.

12.13.1. Переменные

- `BROKEN` предназначена для портов, которые в настоящее время не компилируются, не устанавливаются или не удаляются правильно. Следует использовать, когда проблема считается временной.

  В особых случаях кластер построения будет продолжать попытки собрать их, чтобы показать, решена ли основная проблема. (Однако, как правило, кластер запускается без этой возможности.)

  В частности, используйте `BROKEN`, когда порт:

  - не компилируется
  - не выполняет процесс своей конфигурации или установки
  - устанавливает файлы вовне ${LOCALBASE}
  - не удаляет полностью все свои файлы при деинсталляции (тем не менее, это может быть допустимо, и подходит для портов, оставляющих после себя файлы, измененные пользователем)

- `FORBIDDEN` используется для портов, которые содержат уязвимости в информационной безопасности или являются потенциально вредными в плане обеспечения информационной безопасности системы FreeBSD при установке данного порта (например: заведомо небезопасная программа или программа, которая предоставляет легко взламываемые сервисы). Порты должны помечаться как `FORBIDDEN`, как только в конкретном программном обеспечении обнаружилась уязвимость, но обновление выпущено не было. В идеальном случае порты должны обновляться максимально быстро после обнаружения уязвимости, чтобы уменьшить число уязвимых хостов FreeBSD (нам нравится иметь репутацию безопасной системы), однако иногда случается значительный временной разрыв между обнаружением уязвимости и выходом обновлённого релиза уязвимого программного обеспечения. Не помечайте порт как `FORBIDDEN`, если причина не вызвана соображениями информационной безопасности.

- `IGNORE` предназначена для портов, которые не должны строиться по какой-либо другой причине. Следует использовать для портов, в случае когда проблема считается структурной. Кластер построения ни при каких условиях не будет строить порты, помеченные как `IGNORE`. В частности, используйте `IGNORE`, когда порт:

  - компилируется, но работает неправильно
  - не работает на установленной версии FreeBSD
  - имеет дистрибутивный файл, который не может быть автоматически извлечен из-за лицензионных ограничений
  - не работает с каким-либо другим портом, установленным в настоящее время (например, порт зависит от www/apache20, но установлен www/apache22)

Если порт будет конфликтовать с уже установленным портом (например, если они устанавливают файл в то же место, но с иным функциональным назначением), то используйте вместо этого CONFLICTS. CONFLICTS сам установит значение IGNORE.

- Если порт нужно пометить как IGNORE только на некоторых архитектурах, для этого есть две другие удобные переменные, которые автоматически установят для вас значения: ONLY_FOR_ARCHS и NOT_FOR_ARCHS. Примеры:

```
ONLY_FOR_ARCHS= i386 amd64
```

```
NOT_FOR_ARCHS=  ia64 sparc64
```

  Собственное сообщение IGNORE можно задать с использованием ONLY_FOR_ARCHS_REASON и NOT_FOR_ARCHS_REASON. Отдельно для каждой архитектуры это возможно с использованием ONLY_FOR_ARCHS_REASON_ARCH и NOT_FOR_ARCHS_REASON_ARCH.

- Если порт загружает и устанавливает исполняемые файлы i386, то следует установить IA32_BINARY_PORT. Если эта переменная установлена, будет выполнена проверка доступности каталога /usr/lib32 для библиотек версии IA32 и поддержки IA32 в ядре. При невыполнении любого из этих условий будет автоматически установлена переменная IGNORE.

12.13.2. Замечания по реализации

Строки не следует брать в кавычки. Также построение строки должно несколько различаться из-за способа отображения информации пользователю. Примеры:

```
BROKEN= fails to link with base -lcrypto
```

```
IGNORE= unsupported on recent versions
```

получаемые в результате следующего вывода make describe:

```
===>  foobar-0.1 is marked as broken: fails to link with base -lcrypto.
```

```
===>  foobar-0.1 is unsupported on recent versions.
```

12.14. Пометка порта на удаление с DEPRECATED или EXPIRATION_DATE

Помните, что BROKEN и FORBIDDEN будут использованы как временное средство, если порт не является работающим. Постоянно неработоспособные порты должны полностью удаляться из дерева.

В подходящих ситуациях пользователи могут быть оповещены о предстоящем удалении через переменные DEPRECATED и EXPIRATION_DATE. Первое - это просто строка, сообщающая причину запланированного удаления порта; вторая является строкой в формате ISO 8601 (YYYY-MM-DD). Обе будут показаны пользователю.

Переменную DEPRECATED можно установить без использования EXPIRATION_DATE (в частности, при рекомендации новой версии порта), но обратный порядок не имеет никакого смысла.

Не существует устоявшейся политики, как долго следует продолжать уведомления. Текущая практика дает около месяца для решения проблем безопасности и два месяца для проблем построения. Это также дает немного времени на исправление проблем любым заинтересованным коммиттерам.

12.15. Избегайте использования конструкции .error

Правильным способом подать сигнал для Makefile о том, что порт не может быть установлен из-за какого-то внешнего фактора (например, пользователь указал недопустимую комбинацию опций построения), является установка непустого значения для IGNORE. Это значение будет сформатировано и показано пользователю во время make install.

Использование для этих целей .error является распространенной ошибкой. Проблема в том, что в этой ситуации будут повреждены многие инструменты автоматизации, работающие с деревом портов. Наибольшим образом это распространено при попытке построить /usr/ports/INDEX (смотрите Запуск make describe). Тем не менее, даже более простые команды, такие как make maintainer, в этом случае также вернут ошибку. Это не является приемлемым.

Пример 34. Как избегать использования .error

Из следующих двух вариантов строки файла Makefile первый приведёт к неудачному завершению работы make index, а второй - нет:

```
.error "option is not supported"
```

```
IGNORE=option is not supported
```

12.16. Использование sysctl

Использование sysctl не рекомендуется, кроме как при выполнении целей. Это вызвано тем, что вычисление любых `makevar`, таких как во время команды `make index`, с необходимостью запуска этой команды, еще больше замедляет весь процесс.

sysctl(8) следует всегда использовать через переменную `SYSCTL`, поскольку она содержит полностью заданный путь, и при необходимости может быть переопределена.

12.17. Меняющиеся дистрибутивные файлы

Иногда авторы программного обеспечения меняют содержимое выпущенных дистрибутивных файлов без смены названия. Вы должны проверять, что изменения являются официальными и произведены автором. В прошлом бывало, что дистрибутивный файл молча изменялся на сайтах загрузки с намерением нанести вред или скомпрометировать безопасность конечного пользователя.

Отложите старый файл с дистрибутивом в сторону, загрузите новый, распакуйте его и сравните содержимое при помощи diff(1). Если вы не видите ничего подозрительного, то можете обновить файл distinfo. Убедитесь, что вы подытожили различия в вашем PR или описании коммита, чтобы другие люди были в курсе, что вы позаботились о том, что ничего плохого не случилось.

Возможно вы также захотите связаться с автором этого программного обеспечения для подтверждения изменений.

12.18. Избегание линуксизмов

Не используйте /proc, если доступны любые другие источники получения информации, например, `setprogname(argv[0])` в `main()` и getprogname(3), в случае если вы хотите "знать своё имя".

Не полагайтесь на поведение, не регламентированное POSIX.

Не выполняйте запись временных меток в критических путях выполнения приложения, если можно обойтись без этого. Получение временных меток может быть медленным, в зависимости от степени точности используемых часов в операционной системе. Если временные метки действительно нужны, определите степень требуемой точности и используйте тот API, в котором документируется получение достаточной точности.

Ряд простых системных вызовов (например, gettimeofday(2), getpid(2)) работают намного быстрее в Linux® по сравнению с любой другой операционной системой из-за кеширования и используемой оптимизации vsyscall. Не полагайтесь на их дешевизну в критичных к производительности приложениях. В целом, старайтесь избегать системных вызовов там, где это возможно.

Не полагайтесь на специфичное для Linux® поведение сокета. В частности, отличаются размеры буфера сокета по умолчанию (выполните вызов setsockopt(2) с `SO_SNDBUF` и `SO_RCVBUF`,

и в то время как в Linux® при заполнении буфера сокета send(2) блокируется, FreeBSD возвращает ошибку и устанавливает `ENOBUFS` в качестве значения errno.

Если требуется рассчитывать на нестандартное поведение, инкапсулируйте это должным образом в общий для всех API с проверкой поведения на этапе конфигурации, и если требуемое поведение не найдено, прекращайте выполнение.

Используйте страницы справочника для проверки, относится ли функция к интерфейсу POSIX (ищите раздел "STANDARDS" на странице справочника).

Не рассчитывайте на то, что в качестве /bin/sh используется bash. Убедитесь, что командная строка, переданная в system(3), будет работать в POSIX-совместимой оболочке.

Список основных bash-измов расположен здесь.

Проверьте, что используемые заголовочные файлы включены в POSIX или список, рекомендуемый страницей справочника, т.к. например, забыть подключить sys/types.h - не такая уж проблема в Linux®, однако это не так во FreeBSD.

Компилируйте многопоточные приложения с ключом "-pthread", а не "-lpthread" или как-либо ещё.

12.19. Разное

Файлы pkg-descr и pkg-plist должны проверяться дважды. Если вы пересматриваете порт и думаете, что его можно описать иначе, сделайте это.

Пожалуйста, не создавайте дополнительных копий лицензии GNU General Public License в нашей системе.

Будьте внимательны с юридическими вопросами! Не делайте из нас нелегальных распространителей ПО!

Глава 13. Примерный Makefile

Вот примерный Makefile, который можно использовать при создании нового порта. Обязательно удалите все дополнительные комментарии (те, которые в скобках)!

Вам рекомендуется следовать этому формату (соблюдая порядок следования переменных, пустые строки между разделами, и так далее). Этот формат разработан для того, чтобы важная информация была легко найдена. Мы рекомендуем вам воспользоваться утилитой portlint для проверки файла Makefile.

```
[заголовок...просто чтобы нам было легче идентифицировать порт.]
# Created by: Satoshi Asami <asami@FreeBSD.org>
[Необязательная строка Created by: содержит имя
человека, создавшего первоначальную версию порта.  Следует отметить,
что за : следует пробел, но не символ табуляции.  Если
эта строка присутствует, будущие сопровождающие не должны её менять
или удалять, кроме как по запросу первоначального автора.]

# $FreeBSD$
[ ^^^^^^^^ Эта строка будет автоматически заменена на строчку RCS ID
системой SVN при выполнении операции коммита в наше хранилище.  При
обновлении порта не приводите эту строку обратно к виду
"$FreeBSD$".  SVN сделает это автоматически.]

[секция описания собственно порта и основного сервера - сначала всегда
 PORTNAME и PORTVERSION, за ним следует CATEGORIES, а затем
 MASTER_SITES, за которым может идти MASTER_SITE_SUBDIR.
 PKGNAMEPREFIX и PKGNAMESUFFIX, если они нужны, следуют за ними.
 Затем следует DISTNAME, EXTRACT_SUFX и/или DISTFILES, а потом, если это нужно,
 EXTRACT_ONLY.]
PORTNAME=   xdvi
PORTVERSION=    18.2
CATEGORIES= print
[не забывайте про завершающую косую черту ("/")!
 если вы не используете макросы MASTER_SITE_*]
MASTER_SITES=   ${MASTER_SITE_XCONTRIB}
MASTER_SITE_SUBDIR= applications
PKGNAMEPREFIX=  ja-
DISTNAME=   xdvi-pl18
[задайте это, если исходный код поставляется не в виде
 стандартного файла ".tar.gz"]
EXTRACT_SUFX=   .tar.Z

[секция патчей -- может быть пустой]
PATCH_SITES=    ftp://ftp.sra.co.jp/pub/X11/japanese/
PATCHFILES= xdvi-18.patch1.gz xdvi-18.patch2.gz

[сопровождающий; *обязательное поле*!  Это человек, который добровольно
 занимается обновлениями порта и неисправностями при построении, и которому
```

```
 пользователь может направлять вопросы и сообщения об ошибках.  Для
 сохранения как можно более высокого качества Коллекции Портов мы больше
 не принимаем новые порты, назначенные на "ports@FreeBSD.org".]
MAINTAINER= asami@FreeBSD.org
COMMENT=    DVI Previewer for the X Window System

[зависимости -- могут быть пустыми]
RUN_DEPENDS=    gs:${PORTSDIR}/print/ghostscript

[этот раздел для остальных стандартных переменных из bsd.port.mk, кроме
 тех, что перечислены выше]
[Если порт задает вопросы во время этапов настройки, построения,
 установки...]
IS_INTERACTIVE= yes
[Если распаковка происходит в каталог, отличных от ${DISTNAME}...]
WRKSRC=     ${WRKDIR}/xdvi-new
[Если патчи делались не относительно ${WRKSRC}, вам, может быть, не
 придется изменять эту переменную]
PATCH_DIST_STRIP=   -p1
[Если порт требует скрипта "configure", генерируемого GNU-версией программы
 autoconf]
GNU_CONFIGURE=  yes
[Если для построения порту требуется GNU-версия утилиты make, а не
 /usr/bin/make...]
USES=       gmake
[Если это приложение X и требует запуска "xmkmf -a"...]
USES=       imake
[и так далее]

[В правилах ниже используются нестандартные переменные]
MY_FAVORITE_RESPONSE=   "yeah, right"

[теперь специальные правила, в порядке их вызова]
pre-fetch:
    я что-то выкачиваю, точно

post-patch:
    мне кое-что сделать после применения патча, великолепно

pre-install:
    и потом еще кое-что перед установкой, ого

[и, наконец, эпилог]

.include <bsd.port.mk>
```

Глава 14. Актуализация

Коллекция Портов FreeBSD постоянно изменяется. Здесь находится некоторая информация о том, как поддерживать её в актуальном состоянии.

14.1. FreshPorts

Самым простым способом отслеживать уже произошедшие обновления является подписка на FreshPorts. Для мониторинга вы можете выбрать несколько портов. Мейнтейнерам настоятельно рекомендуется подписаться здесь, потому что они будут получать уведомления не только о собственных изменениях, но и об изменениях, сделанных любым другим коммиттером FreeBSD. (Это часто необходимо для синхронизации с изменениями на более низком технологическом уровне-хотя более корректным было бы получение предупреждений от тех, кто вносит подобные изменения, иногда этот этап пропускается или он просто непрактичен. Кроме того, в некоторых случаях изменения по своей природе весьма незначительны. Мы полагаем, что любой разработчик в таких ситуациях будет руководствоваться здравым смыслом).

Если вы хотите использовать FreshPorts, то вам нужна только учётная запись. Если регистрационный адрес вашей электронной почты будет иметь вид `@FreeBSD.org`, то справа на Web-страницах вы увидите дополнительную ссылку. Для тех из вас, кто уже получил учётную запись FreshPorts, но не использовал собственный адрес электронной почты `@FreeBSD.org`, достаточно сменить адрес на `@FreeBSD.org`, подписаться, а затем сменить его обратно.

Во FreshPorts имеется также функция проверки правильности, которая автоматически проверяет каждое изменение, внесённое в дерево портов FreeBSD. Если вы подпишетесь на эту услугу, то будете оповещаться обо всех ошибках, обнаруженных FreshPorts при проверке внесённых вами изменений.

14.2. Web-интерфейс к хранилищу исходных текстов

Файлы в хранилище исходных текстов можно просматривать при помощи Web-интерфейса. Изменения, которые касаются в целом всей системы портов, теперь документируются в файле CHANGES. Изменения, касающиеся отдельных портов, отражаются теперь в файле UPDATING. Однако однозначный ответ на любой вопрос можно найти, только прочитав исходных код bsd.port.mk и связанных с ним файлов.

14.3. Список рассылки FreeBSD, посвящённый портам

Если вы поддерживаете порты, то должны следить за Список рассылки, посвящённый Портам FreeBSD. О важных изменениях, отражающихся на работе портов, будет сообщаться здесь, а затем они переносятся в CHANGES.

Если данный список рассылки слишком загружен сообщениями, вы можете отслеживать freebsd-ports-announce, который модерируется и не является местом для дискуссий.

14.4. Кластер построения портов FreeBSD

Одной из наименее известных сильных сторон FreeBSD является тот факт, что для непрерывного построения Коллекции Портов для каждого из основных релизов ОС для каждой архитектуры уровня поддержки Tier-1 выделен целый кластер машин.

Отдельные порты собираются, если они специально не помечены как `IGNORE`. Для портов, помеченных как `BROKEN`, попытки будут продолжены для того, чтобы увидеть, если основная проблема была решена. (Это сделано через использование переменной `TRYBROKEN` для Makefile порта.)

14.5. Portscout: сканер дистрибутивных файлов портов FreeBSD

Кластер построения выделен для выполнения самого последнего релиза каждого из портов, дистрибутивные файлы которых уже были сгружены. Однако из-за постоянных изменений в Internet дистрибутивные файлы могут быстро исчезать. Portscout, средство сканирования дистрибутивных файлов FreeBSD пытается опросить каждый из сайтов, доступных для сгрузки каждого из портов, для определения того, доступны ли ещё дистрибутивные файлы. Portscout может готовить отчёты в HTML и рассылать электронные письма об имеющихся обновлениях для портов тем, кто это запрашивает. Мейнтейнеры периодически запрашивают наличие изменений, либо вручную, либо используя ленту RSS.

Главная страница Portscout отображает email мейнтейнера порта, количество портов, за которые ответственен мейнтейнер, количество портов с новыми дистрибутивными файлами и процент устаревших портов. Функция поиска выполняет поиск мейнтейнера по адресу электронной почты и позволяет выбирать между всеми портами или только устаревшими.

При щелчке по адресу электронной почты мейнтейнера отображается список всех его портов, разделённых по категориям, вместе с текущим номером версии, информацией о наличии новой версии, временем последнего обновления порта и временем его последней проверки. Функция поиска на этой странице позволяет пользователю выполнять поиск конкретного порта.

По щелчку на название порта в списке отображается информация о порте FreshPorts.

14.6. Система мониторинга портов FreeBSD

Другим полезным ресурсом является Система мониторинга портов FreeBSD (известная также как `portsmon`). Система представляет собой базу данных, обрабатывающую информацию из нескольких источников и позволяющую просматривать их при помощи Web-интерфейса. На данный момент задействованы база сообщений об ошибках (PR), протоколы ошибок кластера построения и отдельные файлы из коллекции портов. В

будущем в этот список будет добавлена система проверки дистрибутивных файлов и другие ресурсы.

Для начала вы можете просмотреть всю информацию о некотором порте при помощи средства Обзор отдельного порта.

На момент написания это единственный доступный ресурс, который для имени порта ставит в соответствие записи PR GNATS. (Отправители PR не всегда добавляют в название имя порта, хотя мы предпочитаем, чтобы они это делали.) Таким образом, `portsmon` это хорошее место для начала, если вы хотите найти присланные PR и/или ошибки построения для существующего порта; либо поискать, был ли уже прислан новый порт, который вы подумывали создать сами.

Глава 15. Значения USES

15.1. An Introduction to USES

USES macros make it easy to declare requirements and settings for a port. They can add dependencies, change building behavior, add metadata to packages, and so on, all by selecting simple, preset values.

Each section in this chapter describes a possible value for USES, along with its possible arguments. Arguments are appended to the value after a colon (:). Multiple arguments are separated by commas (,).

Пример 35. Using Multiple Values

```
USES=   bison perl
```

Пример 36. Adding an Argument

```
USES=   tar:xz
```

Пример 37. Adding Multiple Arguments

```
USES=   drupal:7,theme
```

Пример 38. Mixing it All Together

```
USES=   pgsql:9.3+ cpe python:2.7,build
```

15.2. 7z

Possible arguments: (none), p7zip, partial

Extract using 7z(1) instead of bsdtar(1) and sets EXTRACT_SUFX=.7z. The p7zip option forces a dependency on the 7z from archivers/p7zip if the one from the base system is not able to extract the files. EXTRACT_SUFX is not changed if the partial option is used, this can be used if the main distribution file does not have a .7z extension.

15.3. ada

Possible arguments: (none), 5, 6

Depends on an Ada-capable compiler, and sets CC accordingly. Defaults to use gcc 5 from ports. Use the :_X_ version option to force building with a different version.

15.4. autoreconf

Possible arguments: (none), build

Runs autoreconf. It encapsulates the aclocal, autoconf, autoheader, automake, autopoint, and libtoolize commands. Each command applies to ${AUTORECONF_WRKSRC}/configure.ac or its old name, ${AUTORECONF_WRKSRC}/configure.in. If configure.ac defines subdirectories with their own configure.ac using AC_CONFIG_SUBDIRS, autoreconf will recursively update those as well. The :build argument only adds build time dependencies on those tools but does not run autoreconf. A port can set AUTORECONF_WRKSRC if WRKSRC does not contain the path to configure.ac.

15.5. blaslapack

Possible arguments: (none), atlas, netlib (default), gotoblas, openblas

Adds dependencies on Blas / Lapack libraries.

15.6. bdb

Possible arguments: (none), 48, 5 (default), 6

Add dependency on the Berkeley DB library. Default to databases/db5. It can also depend on databases/db48 when using the :48 argument or databases/db6 with :6. It is possible to declare a range of acceptable values, :48+ finds the highest installed version, and falls back to 4.8 if nothing else is installed. INVALID_BDB_VER can be used to specify versions which do not work with this port. The framework exposes the following variables to the port:

BDB_LIB_NAME

The name of the Berkeley DB library. For example, when using databases/db5, it contains db-5.3.

BDB_LIB_CXX_NAME

The name of the Berkeley DBC++ library. For example, when using databases/db5, it contains db_cxx-5.3.

BDB_INCLUDE_DIR

The location of the Berkeley DB include directory. For example, when using databases/db5, it will contain ${LOCALBASE}/include/db5.

BDB_LIB_DIR

The location of the Berkeley DB library directory. For example, when using databases/db5, it contains ${LOCALBASE}/lib.

`BDB_VER`

The detected Berkeley DB version. For example, if using `USES=bdb:48+` and Berkeley DB 5 is installed, it contains `5`.

> **!** databases/db48 is deprecated and unsupported. It must not be used by any port.

15.7. `bison`

Possible arguments: (none), `build`, `run`, `both`

Uses devel/bison By default, with no arguments or with the `build` argument, it implies `bison` is a build-time dependency, `run` implies a run-time dependency, and `both` implies both run-time and build-time dependencies.

15.8. `cabal`

> **!** Ports should not be created for Haskell libraries, see Haskell Libraries for more information.

Possible arguments: (none), `hpack`

Sets default values and targets used to build Haskell software using Cabal. A build dependency on the Haskell compiler port (GHC) is added. If `hpack` argument is given, a build dependency on devel/hs-hpack is added and `hpack` is invoked at configuration step to generate. cabal file.

The framework provides the following variables:

`USE_CABAL`

If the software uses Haskell dependencies, list them in this variable. Each item should be present on Hackage and be listed in form `packagename-0.1.2`. Dependencies can have revisions, which are specified after the _ symbol. Automatic generation of dependency list is supported, see Building Haskell Applications with `cabal`.

`CABAL_FLAGS`

List of flags to be passed to `cabal-install` during the configuring and building stage. The flags are passed verbatim.

`EXECUTABLES`

List of executable files installed by the port. Default value: `${PORTNAME}`. Items from this list are automatically added to pkg-plist.

`SKIP_CABAL_PLIST`

If defined, do not add items from `${EXECUTABLES}` to pkg-plist.

`opt_USE_CABAL`

Adds items to `${USE_CABAL}` depending on `opt` option.

`opt_EXECUTABLES`

Adds items to `${EXECUTABLES}` depending on `opt` option.

`opt_CABAL_FLAGS`

If `opt` is enabled, append the value to `${CABAL_FLAGS}`. Otherwise, append `-value` to disable the flag.

`FOO_DATADIR_VARS`

For an executable named `FOO` list Haskell packages, whose data files should be accessible by the executable.

15.9. `cargo`

Possible arguments: (none)

Uses Cargo for configuring, building, and testing. It can be used to port Rust applications that use the Cargo build system. For more information see Building Rust Applications with `cargo`.

15.10. `charsetfix`

Possible arguments: (none)

Prevents the port from installing charset.alias. This must be installed only by converters/libiconv. `CHARSETFIX_MAKEFILEIN` can be set to a path relative to `WRKSRC` if charset.alias is not installed by ${WRKSRC}/Makefile.in.

15.11. `cmake`

Possible arguments: (none), `insource`, `noninja`, `run`, `testing`

Use CMake for configuring the port and generating a build system.

By default an out-of-source build is performed, leaving the sources in `WRKSRC` free from build artifacts. With the `insource` argument, an in-source build will be performed instead. This argument should be an exception, used only when a regular out-of-source build does not work.

By default Ninja (devel/ninja) is used for the build. In some cases this does not work correctly. With the `noninja` argument, the build will use regular `make` for builds. This argument should only be used if a Ninja-based build does not work.

With the `run` argument, a run dependency is registered in addition to a build dependency.

With the `testing` argument, a test-target is added that uses CTest. When running tests the port will be re-configured for testing and re-built.

For more information see Using `cmake`.

15.12. compiler

Possible arguments: (none), `env` (default, implicit), `c++17-lang`, `c++14-lang`, `c++11-lang`, `gcc-c++11-lib`, `c++11-lib`, `c++0x`, `c11`, `nestedfct`, `features`

Determines which compiler to use based on any given wishes. Use `c++17-lang` if the port needs a c++17-capable compiler, `c++14-lang` if the port needs a c++14-capable compiler, `c++11-lang` if the port needs a c++11-capable compiler, `gcc-c++11-lib` if the port needs the `g++` compiler with a c++11 library, or `c++11-lib` if the port needs a c++11-ready standard library. If the port needs a compiler understanding c++0X, C11 or nested functions, the corresponding parameters should be used.

Use `features` to request a list of features supported by the default compiler. After including bsd.port.pre.mk the port can inspect the results using these variables:

- `COMPILER_TYPE`: the default compiler on the system, either gcc or clang
- `ALT_COMPILER_TYPE`: the alternative compiler on the system, either gcc or clang. Only set if two compilers are present in the base system.
- `COMPILER_VERSION`: the first two digits of the version of the default compiler.
- `ALT_COMPILER_VERSION`: the first two digits of the version of the alternative compiler, if present.
- `CHOSEN_COMPILER_TYPE`: the chosen compiler, either gcc or clang
- `COMPILER_FEATURES`: the features supported by the default compiler. It currently lists the c++ library.

15.13. cpe

Possible arguments: (none)

Include Common Platform Enumeration (CPE) information in package manifest as a CPE 2.3 formatted string. See the CPE specification for details. To add CPE information to a port, follow these steps:

1. Search for the official CPE entry for the software product either by using the NVD's CPE search engine or in the official CPE dictionary (warning, very large XML file). *Do not ever make up CPE data.*
2. Add `cpe` to `USES` and compare the result of `make -V CPE_STR` to the CPE dictionary entry. Continue one step at a time until `make -V CPE_STR` is correct.
3. If the product name (second field, defaults to `PORTNAME`) is incorrect, define `CPE_PRODUCT`.
4. If the vendor name (first field, defaults to `CPE_PRODUCT`) is incorrect, define `CPE_VENDOR`.
5. If the version field (third field, defaults to `PORTVERSION`) is incorrect, define `CPE_VERSION`.
6. If the update field (fourth field, defaults to empty) is incorrect, define `CPE_UPDATE`.
7. If it is still not correct, check Mk/Uses/cpe.mk for additional details, or contact the Ports Security Team <ports-secteam@FreeBSD.org>.
8. Derive as much as possible of the CPE name from existing variables such as `PORTNAME` and `PORTVERSION`. Use variable modifiers to extract the relevant portions from these variables rather

than hardcoding the name.

9. *Always* run `make -V CPE_STR` and check the output before committing anything that changes `PORTNAME` or `PORTVERSION` or any other variable which is used to derive `CPE_STR`.

15.14. `cran`

Possible arguments: (none), `auto-plist`, `compiles`

Uses the Comprehensive R Archive Network. Specify `auto-plist` to automatically generate pkg-plist. Specify `compiles` if the port has code that need to be compiled.

15.15. `desktop-file-utils`

Possible arguments: (none)

Uses update-desktop-database from devel/desktop-file-utils. An extra post-install step will be run without interfering with any post-install steps already in the port Makefile. A line with `@desktop-file-utils` will be added to the plist.

15.16. `desthack`

Possible arguments: (none)

Changes the behavior of GNU configure to properly support `DESTDIR` in case the original software does not.

15.17. `display`

Possible arguments: (none), *ARGS*

Set up a virtual display environment. If the environment variable `DISPLAY` is not set, then Xvfb is added as a build dependency, and `CONFIGURE_ENV` is extended with the port number of the currently running instance of Xvfb. The *ARGS* parameter defaults to `install` and controls the phase around which to start and stop the virtual display.

15.18. `dos2unix`

Possible arguments: (none)

The port has files with line endings in DOS format which need to be converted. Several variables can be set to control which files will be converted. The default is to convert *all* files, including binaries. See Simple Automatic Replacements for examples.

- `DOS2UNIX_REGEX`: match file names based on a regular expression.
- `DOS2UNIX_FILES`: match literal file names.
- `DOS2UNIX_GLOB`: match file names based on a glob pattern.

- DOS2UNIX_WRKSRC: the directory from which to start the conversions. Defaults to ${WRKSRC}.

15.19. drupal

Possible arguments: 7, module, theme

Automate installation of a port that is a Drupal theme or module. Use with the version of Drupal that the port is expecting. For example, USES=drupal:7,module says that this port creates a Drupal 6 module. A Drupal 7 theme can be specified with USES=drupal:7,theme.

15.20. eigen

Possible arguments: 2, 3, build (default), run

Add dependency on math/eigen.

15.21. fakeroot

Possible arguments: (none)

Changes some default behavior of build systems to allow installing as a user. See https://wiki.debian.org/FakeRoot for more information on fakeroot.

15.22. fam

Possible arguments: (none), fam, gamin

Uses a File Alteration Monitor as a library dependency, either devel/fam or devel/gamin. End users can set WITH_FAM_SYSTEM to specify their preference.

15.23. firebird

Possible arguments: (none), 25

Add a dependency to the client library of the Firebird database.

15.24. fonts

Possible arguments: (none), fc, fcfontsdir (default), fontsdir, none

Adds a runtime dependency on tools needed to register fonts. Depending on the argument, add a @fc ${FONTSDIR} line, @fcfontsdir ${FONTSDIR} line, @fontsdir ${FONTSDIR} line, or no line if the argument is none, to the plist. FONTSDIR defaults to ${PREFIX}/share/fonts/${FONTNAME} and FONTNAME to ${PORTNAME}. Add FONTSDIR to PLIST_SUB and SUB_LIST

15.25. fortran

Possible arguments: gcc (default)

Uses the GNU Fortran compiler.

15.26. fuse

Possible arguments: 2 (default), 3

The port will depend on the FUSE library and handle the dependency on the kernel module depending on the version of FreeBSD.

15.27. gem

Possible arguments: (none), noautoplist

Handle building with RubyGems. If noautoplist is used, the packing list is not generated automatically.

15.28. gettext

Possible arguments: (none)

Deprecated. Will include both gettext-runtime and gettext-tools.

15.29. gettext-runtime

Possible arguments: (none), lib (default), build, run

Uses devel/gettext-runtime. By default, with no arguments or with the lib argument, implies a library dependency on libintl.so. build and run implies, respectively a build-time and a run-time dependency on gettext.

15.30. gettext-tools

Possible arguments: (none), build (default), run

Uses devel/gettext-tools. By default, with no argument, or with the build argument, a build time dependency on msgfmt is registered. With the run argument, a run-time dependency is registered.

15.31. ghostscript

Possible arguments: *X*, build, run, nox11

A specific version *X* can be used. Possible versions are 7, 8, 9, and agpl (default). nox11 indicates that the -nox11 version of the port is required. build and run add build- and run-time dependencies on

Ghostscript. The default is both build- and run-time dependencies.

15.32. gl

Possible arguments: (none)

Provides an easy way to depend on GL components. The components should be listed in `USE_GL`. The available components are:

`egl`
: add a library dependency on libEGL.so from graphics/libglvnd

`gbm`
: Add a library dependency on libgbm.so from graphics/mesa-libs

`gl`
: Add a library dependency on libGL.so from graphics/libglvnd

`glesv2`
: Add a library dependency on libGLESv2.so from graphics/libglvnd

`glew`
: Add a library dependency on libGLEW.so from graphics/glew

`glu`
: Add a library dependency on libGLU.so from graphics/libGLU

`glut`
: Add a library dependency on libglut.so from graphics/freeglut

`opengl`
: Add a library dependency on libOpenGL.so from graphics/libglvnd

15.33. gmake

Possible arguments: (none)

Uses devel/gmake as a build-time dependency and sets up the environment to use `gmake` as the default `make` for the build.

15.34. gnome

Possible arguments: (none)

Provides an easy way to depend on GNOME components. The components should be listed in `USE_GNOME`. The available components are:

- `atk`

- atkmm
- cairo
- cairomm
- dconf
- esound
- evolutiondataserver3
- gconf2
- gconfmm26
- gdkpixbuf
- gdkpixbuf2
- glib12
- glib20
- glibmm
- gnomecontrolcenter3
- gnomedesktop3
- gnomedocutils
- gnomemenus3
- gnomemimedata
- gnomeprefix
- gnomesharp20
- gnomevfs2
- gsound
- gtk-update-icon-cache
- gtk12
- gtk20
- gtk30
- gtkhtml3
- gtkhtml4
- gtkmm20
- gtkmm24
- gtkmm30
- gtksharp20
- gtksourceview
- gtksourceview2
- gtksourceview3

- gtksourceviewmm3
- gvfs
- intlhack
- intltool
- introspection
- libartlgpl2
- libbonobo
- libbonoboui
- libgda5
- libgda5-ui
- libgdamm5
- libglade2
- libgnome
- libgnomecanvas
- libgnomekbd
- libgnomeprint
- libgnomeprintui
- libgnomeui
- libgsf
- libgtkhtml
- libgtksourceviewmm
- libidl
- librsvg2
- libsigc++12
- libsigc++20
- libwnck
- libwnck3
- libxml++26
- libxml2
- libxslt
- metacity
- nautilus3
- orbit2
- pango
- pangomm

- pangox-compat
- py3gobject3
- pygnome2
- pygobject
- pygobject3
- pygtk2
- pygtksourceview
- referencehack
- vte
- vte3

The default dependency is build- and run-time, it can be changed with `:build` or `:run`. For example:

```
USES=       gnome
USE_GNOME=  gnomemenus3:build intlhack
```

See Using GNOME for more information.

15.35. go

Ports should not be created for Go libs, see Go Libraries for more information.

Possible arguments: (none), `modules`, `no_targets`, `run`

Sets default values and targets used to build Go software. A build dependency on the Go compiler port selected via `GO_PORT` is added. By default the build is performed in GOPATH mode. If Go software uses modules, the modules-aware mode can be switched on with `modules` argument. `no_targets` will setup build environment like `GO_ENV`, `GO_BUILDFLAGS` but skip creating `post-extract` and `do-{build,install,test}` targets. `run` will also add a run dependency on what is in `GO_PORT`.

The build process is controlled by several variables:

`GO_MODULE`

The name of the application module as specified by the `module` directive in `go.mod`. In most cases, this is the only required variable for ports that use Go modules.

`GO_PKGNAME`

The name of the Go package when building in GOPATH mode. This is the directory that will be created in `${GOPATH}/src`. If not set explicitly and `GH_SUBDIR` or `GL_SUBDIR` is present, `GO_PKGNAME` will be inferred from it. It is not needed when building in modules-aware mode.

`GO_TARGET`

The packages to build. The default value is `${GO_PKGNAME}`. `GO_TARGET` can also be a tuple in the form `package:path` where path can be either a simple filename or a full path starting with

${PREFIX}.

GO_TESTTARGET

The packages to test. The default value is ./⋯ (the current package and all subpackages).

CGO_CFLAGS

Additional CFLAGS values to be passed to the C compiler by go.

CGO_LDFLAGS

Additional LDFLAGS values to be passed to the C compiler by go.

GO_BUILDFLAGS

Additional build arguments to be passed to go build.

GO_TESTFLAGS

Additional build arguments to be passed to go test.

GO_PORT

The Go compiler port to use. By default this is lang/go but can be set to lang/go-devel in make.conf for testing with future Go versions.

This variable must not be set by individual ports!

See Building Go Applications for usage examples.

15.36. gperf

Possible arguments: (none)

Add a buildtime dependency on devel/gperf if gperf is not present in the base system.

15.37. grantlee

Possible arguments: 5, selfbuild

Handle dependency on Grantlee. Specify 5 to depend on the Qt5 based version, devel/grantlee5. selfbuild is used internally by devel/grantlee5 to get their versions numbers.

15.38. groff

Possible arguments: build, run, both

Registers a dependency on textproc/groff if not present in the base system.

15.39. gssapi

Possible arguments: (none), base (default), heimdal, mit, flags, bootstrap

Handle dependencies needed by consumers of the GSS-API. Only libraries that provide the Kerberos mechanism are available. By default, or set to `base`, the GSS-API library from the base system is used. Can also be set to `heimdal` to use security/heimdal, or `mit` to use security/krb5.

When the local Kerberos installation is not in `LOCALBASE`, set `HEIMDAL_HOME` (for `heimdal`) or `KRB5_HOME` (for `krb5`) to the location of the Kerberos installation.

These variables are exported for the ports to use:

- `GSSAPIBASEDIR`
- `GSSAPICPPFLAGS`
- `GSSAPIINCDIR`
- `GSSAPILDFLAGS`
- `GSSAPILIBDIR`
- `GSSAPILIBS`
- `GSSAPI_CONFIGURE_ARGS`

The `flags` option can be given alongside `base`, `heimdal`, or `mit` to automatically add `GSSAPICPPFLAGS`, `GSSAPILDFLAGS`, and `GSSAPILIBS` to `CFLAGS`, `LDFLAGS`, and `LDADD`, respectively. For example, use `base,flags`.

The `bootstrap` option is a special prefix only for use by security/krb5 and security/heimdal. For example, use `bootstrap,mit`.

Пример 39. Typical Use

```
OPTIONS_SINGLE= GSSAPI
OPTIONS_SINGLE_GSSAPI=  GSSAPI_BASE GSSAPI_HEIMDAL GSSAPI_MIT GSSAPI_NONE

GSSAPI_BASE_USES=   gssapi
GSSAPI_BASE_CONFIGURE_ON=   --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_HEIMDAL_USES=    gssapi:heimdal
GSSAPI_HEIMDAL_CONFIGURE_ON=    --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_MIT_USES=    gssapi:mit
GSSAPI_MIT_CONFIGURE_ON=    --with-gssapi=${GSSAPIBASEDIR}
${GSSAPI_CONFIGURE_ARGS}
GSSAPI_NONE_CONFIGURE_ON=   --without-gssapi
```

15.40. horde

Possible arguments: (none)

Add buildtime and runtime dependencies on devel/pear-channel-horde. Other Horde dependencies

can be added with USE_HORDE_BUILD and USE_HORDE_RUN. See Horde Modules for more information.

15.41. iconv

Possible arguments: (none), lib, build, patch, translit, wchar_t

Uses iconv functions, either from the port converters/libiconv as a build-time and run-time dependency, or from the base system. By default, with no arguments or with the lib argument, implies iconv with build-time and run-time dependencies. build implies a build-time dependency, and patch implies a patch-time dependency. If the port uses the WCHAR_T or //TRANSLIT iconv extensions, add the relevant arguments so that the correct iconv is used. For more information see Using iconv.

15.42. imake

Possible arguments: (none), env, notall, noman

Add devel/imake as a build-time dependency and run xmkmf -a during the configure stage. If the env argument is given, the configure target is not set. If the -a flag is a problem for the port, add the notall argument. If xmkmf does not generate a install.man target, add the noman argument.

15.43. kde

Possible arguments: 5

Add dependency on KDE components. See Using KDE for more information.

15.44. kmod

Possible arguments: (none), debug

Fills in the boilerplate for kernel module ports, currently:

- Add kld to CATEGORIES.
- Set SSP_UNSAFE.
- Set IGNORE if the kernel sources are not found in SRC_BASE.
- Define KMODDIR to /boot/modules by default, add it to PLIST_SUB and MAKE_ENV, and create it upon installation. If KMODDIR is set to /boot/kernel, it will be rewritten to /boot/modules. This prevents breaking packages when upgrading the kernel due to /boot/kernel being renamed to /boot/kernel.old in the process.
- Handle cross-referencing kernel modules upon installation and deinstallation, using @kld.
- If the debug argument is given, the port can install a debug version of the module into KERN_DEBUGDIR/KMODDIR. By default, KERN_DEBUGDIR is copied from DEBUGDIR and set to /usr/lib/debug. The framework will take care of creating and removing any required directories.

15.45. `lha`

Possible arguments: (none)

Set `EXTRACT_SUFX` to `.lzh`

15.46. `libarchive`

Possible arguments: (none)

Registers a dependency on archivers/libarchive. Any ports depending on libarchive must include `USES=libarchive`.

15.47. `libedit`

Possible arguments: (none)

Registers a dependency on devel/libedit. Any ports depending on libedit must include `USES=libedit`.

15.48. `libtool`

Possible arguments: (none), `keepla`, `build`

Patches `libtool` scripts. This must be added to all ports that use `libtool`. The `keepla` argument can be used to keep .la files. Some ports do not ship with their own copy of libtool and need a build time dependency on devel/libtool, use the `:build` argument to add such dependency.

15.49. `linux`

Possible arguments: `c6`, `c7`

Ports Linux compatibility framework. Specify `c6` to depend on CentOS 6 packags. Specify `c7` to depend on CentOS 7 packages. The available packages are:

- `allegro`
- `alsa-plugins-oss`
- `alsa-plugins-pulseaudio`
- `alsalib`
- `atk`
- `avahi-libs`
- `base`
- `cairo`
- `cups-libs`
- `curl`

- cyrus-sasl2
- dbusglib
- dbuslibs
- devtools
- dri
- expat
- flac
- fontconfig
- gdkpixbuf2
- gnutls
- graphite2
- gtk2
- harfbuzz
- jasper
- jbigkit
- jpeg
- libasyncns
- libaudiofile
- libelf
- libgcrypt
- libgfortran
- libgpg-error
- libmng
- libogg
- libpciaccess
- libsndfile
- libsoup
- libssh2
- libtasn1
- libthai
- libtheora
- libv4l
- libvorbis
- libxml2
- mikmod

- naslibs
- ncurses-base
- nspr
- nss
- openal
- openal-soft
- openldap
- openmotif
- openssl
- pango
- pixman
- png
- pulseaudio-libs
- qt
- qt-x11
- qtwebkit
- scimlibs
- sdl12
- sdlimage
- sdlmixer
- sqlite3
- tcl85
- tcp_wrappers-libs
- tiff
- tk85
- ucl
- xorglibs

15.50. localbase

Possible arguments: (none), ldflags

Ensures that libraries from dependencies in LOCALBASE are used instead of the ones from the base system. Specify ldflags to add -L${LOCALBASE}/lib to LDFLAGS instead of LIBS. Ports that depend on libraries that are also present in the base system should use this. It is also used internally by a few other USES.

15.51. lua

Possible arguments: (none), XY, XY+, -XY, XY-ZA, module, flavors, build, run, env

Adds a dependency on Lua. By default this is a library dependency, unless overridden by the build and/or run option. The env option prevents the addition of any dependency, while still defining all the usual variables.

The default version is set by the usual DEFAULT_VERSIONS mechanism, unless a version or range of versions is specified as an argument, for example, 51 or 51-53.

Applications using Lua are normally built for only a single Lua version. However, library modules intended to be loaded by Lua code should use the module option to build with multiple flavors.

For more information see Using Lua.

15.52. lxqt

Possible arguments: (none)

Handle dependencies for the LXQt Desktop Environment. Use USE_LXQT to select the components needed for the port. See Using LXQt for more information.

15.53. makeinfo

Possible arguments: (none)

Add a build-time dependency on makeinfo if it is not present in the base system.

15.54. makeself

Possible arguments: (none)

Indicates that the distribution files are makeself archives and sets the appropriate dependencies.

15.55. mate

Possible arguments: (none)

Provides an easy way to depend on MATE components. The components should be listed in USE_MATE. The available components are:

- autogen
- caja
- common
- controlcenter
- desktop

- dialogs
- docutils
- icontheme
- intlhack
- intltool
- libmatekbd
- libmateweather
- marco
- menus
- notificationdaemon
- panel
- pluma
- polkit
- session
- settingsdaemon

The default dependency is build- and run-time, it can be changed with `:build` or `:run`. For example:

```
USES=       mate
USE_MATE=   menus:build intlhack
```

15.56. meson

Possible arguments: (none)

Provide support for Meson based projects. For more information see Using `meson`.

15.57. metaport

Possible arguments: (none)

Sets the following variables to make it easier to create a metaport: `MASTER_SITES`, `DISTFILES`, `EXTRACT_ONLY`, `NO_BUILD`, `NO_INSTALL`, `NO_MTREE`, `NO_ARCH`.

15.58. mysql

Possible arguments: (none), `version`, `client` (default), `server`, `embedded`

Provide support for MySQL If no version is given, try to find the current installed version. Fall back to the default version, MySQL-5.6. The possible versions are `55`, `55m`, `55p`, `56`, `56p`, `56w`, `57`, `57p`, `80`, `100m`, `101m`, and `102m`. The `m` and `p` suffixes are for the MariaDB and Percona variants of MySQL. `server` and

`embedded` add a build- and run-time dependency on the MySQL server. When using `server` or `embedded`, add `client` to also add a dependency on libmysqlclient.so. A port can set `IGNORE_WITH_MYSQL` if some versions are not supported.

The framework sets `MYSQL_VER` to the detected MySQL version.

15.59. `mono`

Possible arguments: (none), `nuget`

Adds a dependency on the Mono (currently only C#) framework by setting the appropriate dependencies.

Specify `nuget` when the port uses nuget packages. `NUGET_DEPENDS` needs to be set with the names and versions of the nuget packages in the format `name=version`. An optional package origin can be added using `name=version:_origin_`.

The helper target, `buildnuget`, will output the content of the `NUGET_DEPENDS` based on the provided packages.config.

15.60. `motif`

Possible arguments: (none)

Uses x11-toolkits/open-motif as a library dependency. End users can set `WANT_LESSTIF` for the dependency to be on x11-toolkits/lesstif instead of x11-toolkits/open-motif.

15.61. `ncurses`

Possible arguments: (none), `base`, `port`

Uses ncurses, and causes some useful variables to be set.

15.62. `ninja`

Possible arguments: (none)

Uses ninja to build the port.

15.63. `objc`

Possible arguments: (none)

Add objective C dependencies (compiler, runtime library) if the base system does not support it.

15.64. `openal`

Possible arguments: `al`, `soft` (default), `si`, `alut`

Uses OpenAL. The backend can be specified, with the software implementation as the default. The user can specify a preferred backend with `WANT_OPENAL`. Valid values for this knob are `soft` (default) and `si`.

15.65. `pathfix`

Possible arguments: (none)

Look for Makefile.in and configure in `PATHFIX_WRKSRC` (defaults to `WRKSRC`) and fix common paths to make sure they respect the FreeBSD hierarchy. For example, it fixes the installation directory of `pkgconfig's .pc files to ${PREFIX}/libdata/pkgconfig. If the port uses `USES=autoreconf`, Makefile.am will be added to `PATHFIX_MAKEFILEIN` automatically.

If the port `USES=cmake` it will look for CMakeLists.txt in `PATHFIX_WRKSRC`. If needed, that default filename can be changed with `PATHFIX_CMAKELISTSTXT`.

15.66. `pear`

Possible arguments: `env`

Adds a dependency on devel/pear. It will setup default behavior for software using the PHP Extension and Application Repository. Using the `env` arguments only sets up the PEAR environment variables. See PEAR Modules for more information.

15.67. `perl5`

Possible arguments: (none)

Depends on Perl. The configuration is done using `USE_PERL5`.

`USE_PERL5` can contain the phases in which to use Perl, can be `extract`, `patch`, `build`, `run`, or `test`.

`USE_PERL5` can also contain `configure`, `modbuild`, or `modbuildtiny` when Makefile.PL, Build.PL, or Module::Build::Tiny's flavor of Build.PL is required.

`USE_PERL5` defaults to `build run`. When using `configure`, `modbuild`, or `modbuildtiny`, `build` and `run` are implied.

See Using Perl for more information.

15.68. `pgsql`

Possible arguments: (none), `X.Y`, `X.Y+`, `X.Y-`, `X.Y-Z.A`

Provide support for PostgreSQL. Port maintainer can set version required. Minimum and

maximum versions or a range can be specified; for example, `9.0-`, `8.4+`, `8.4-9.2`.

By default, the added dependency will be the client, but if the port requires additional components, this can be done using `WANT_PGSQL=component[:target]`; for example, `WANT_PGSQL=server:configure pltcl plperl`. The available components are:

- `client`
- `contrib`
- `docs`
- `pgtcl`
- `plperl`
- `plpython`
- `pltcl`
- `server`

15.69. `php`

Possible arguments: (none), `phpize`, `ext`, `zend`, `build`, `cli`, `cgi`, `mod`, `web`, `embed`, `pecl`, `flavors`, `noflavors`

Provide support for PHP. Add a runtime dependency on the default PHP version, lang/php56.

`phpize`
: Use to build a PHP extension. Enables flavors.

`ext`
: Use to build, install and register a PHP extension. Enables flavors.

`zend`
: Use to build, install and register a Zend extension. Enables flavors.

`build`
: Set PHP also as a build-time dependency.

`cli`
: Needs the CLI version of PHP.

`cgi`
: Needs the CGI version of PHP.

`mod`
: Needs the Apache module for PHP.

`web`
: Needs the Apache module or the CGI version of PHP.

`embed`

Needs the embedded library version of PHP.

`pecl`

Provide defaults for fetching PHP extensions from the PECL repository. Enables flavors.

`flavors`

Enable automatic PHP flavors generation. Flavors will be generated for all PHP versions, except the ones present in `IGNORE_WITH_PHP`.

`noflavors`

Disable automatic PHP flavors generation. *Must only* be used with extensions provided by PHP itself.

Variables are used to specify which PHP modules are required, as well as which version of PHP are supported.

`USE_PHP`

The list of required PHP extensions at run-time. Add `:build` to the extension name to add a build-time dependency. Example: `pcre xml:build gettext`

`IGNORE_WITH_PHP`

The port does not work with PHP of the given version. For possible values look at the content of `_ALL_PHP_VERSIONS` in Mk/Uses/php.mk.

When building a PHP or Zend extension with `:ext` or `:zend`, these variables can be set:

`PHP_MODNAME`

The name of the PHP or Zend extension. Default value is `${PORTNAME}`.

`PHP_HEADER_DIRS`

A list of subdirectories from which to install header files. The framework will always install the header files that are present in the same directory as the extension.

`PHP_MOD_PRIO`

The priority at which to load the extension. It is a number between `00` and `99`.

For extensions that do not depend on any extension, the priority is automatically set to `20`, for extensions that depend on another extension, the priority is automatically set to `30`. Some extensions may need to be loaded before every other extension, for example www/php56-opcache. Some may need to be loaded after an extension with a priority of `30`. In that case, add `PHP_MOD_PRIO=XX` in the port's Makefile. For example:

```
USES=       php:ext
USE_PHP=    wddx
PHP_MOD_PRIO=   40
```

These variables are available to use in `PKGNAMEPREFIX` or `PKGNAMESUFFIX`:

`PHP_PKGNAMEPREFIX`

Contains `php_XY_-` where *XY* is the current flavor's PHP version. Use with PHP extensions and modules.

`PHP_PKGNAMESUFFIX`

Contains `-php_XY_` where *XY* is the current flavor's PHP version. Use with PHP applications.

`PECL_PKGNAMEPREFIX`

Contains `php_XY_-pecl-` where *XY* is the current flavor's PHP version. Use with PECL modules.

> With flavors, all PHP extensions, PECL extensions, PEAR modules *must have* a different package name, so they must all use one of these three variables in their `PKGNAMEPREFIX` or `PKGNAMESUFFIX`.

15.70. `pkgconfig`

Possible arguments: (none), `build` (default), `run`, `both`

Uses devel/pkgconf. With no arguments or with the `build` argument, it implies `pkg-config` as a build-time dependency. `run` implies a run-time dependency and `both` implies both run-time and build-time dependencies.

15.71. `pure`

Possible arguments: (none), `ffi`

Uses lang/pure. Largely used for building related pure ports. With the `ffi` argument, it implies devel/pure-ffi as a run-time dependency.

15.72. `pyqt`

Possible arguments: (none), `4`, `5`

Uses PyQt. If the port is part of PyQT itself, set `PYQT_DIST`. Use `USE_PYQT` to select the components the port needs. The available components are:

- `core`
- `dbus`
- `dbussupport`
- `demo`
- `designer`
- `designerplugin`
- `doc`
- `gui`

- multimedia
- network
- opengl
- qscintilla2
- sip
- sql
- svg
- test
- webkit
- xml
- xmlpatterns

These components are only available with PyQT4:

- assistant
- declarative
- help
- phonon
- script
- scripttools

These components are only available with PyQT5:

- multimediawidgets
- printsupport
- qml
- serialport
- webkitwidgets
- widgets

The default dependency for each component is build- and run-time, to select only build or run, add `_build` or `_run` to the component name. For example:

```
USES=       pyqt
USE_PYQT=   core doc_build designer_run
```

15.73. python

Possible arguments: (none), `X.Y`, `X.Y+`, `-X.Y`, `X.Y-Z.A`, `patch`, `build`, `run`, `test`

Uses Python. A supported version or version range can be specified. If Python is only needed at build time, run time or for the tests, it can be set as a build, run or test dependency with `build`, `run`, or `test`. If Python is also needed during the patch phase, use `patch`. See Using Python for more information.

`PYTHON_NO_DEPENDS=yes` can be used when the variables exported by the framework are needed but a dependency on Python is not. It can happen when using with `USES=shebangfix`, and the goal is only to fix the shebangs but not add a dependency on Python.

15.74. `qmail`

Possible arguments: (none), `build`, `run`, `both`, `vars`

Uses mail/qmail. With the `build` argument, it implies `qmail` as a build-time dependency. `run` implies a run-time dependency. Using no argument or the `both` argument implies both run-time and build-time dependencies. `vars` will only set QMAIL variables for the port to use.

15.75. `qmake`

Possible arguments: (none), `norecursive`, `outsource`, `no_env`, `no_configure`

Uses QMake for configuring. For more information see Using `qmake`.

15.76. `qt`

Possible arguments: `5`, `no_env`

Add dependency on Qt components. `no_env` is passed directly to `USES= qmake`. See Using Qt for more information.

15.77. `qt-dist`

Possible arguments: (none) or `5` and (none) or one of `3d`, `activeqt`, `androidextras`, `base`, `canvas3d`, `charts`, `connectivity`, `datavis3d`, `declarative`, `doc`, `gamepad`, `graphicaleffects`, `imageformats`, `location`, `macextras`, `multimedia`, `networkauth`, `purchasing`, `quickcontrols2`, `quickcontrols`, `remoteobjects`, `script`, `scxml`, `sensors`, `serialbus`, `serialport`, `speech`, `svg`, `tools`, `translations`, `virtualkeyboard`, `wayland`, `webchannel`, `webengine`, `websockets`, `webview`, `winextras`, `x11extras`, `xmlpatterns`

Provides support for building Qt 5 components. It takes care of setting up the appropriate configuration environment for the port to build.

Пример 40. Building Qt 5 Components

The port is Qt 5's `networkauth` component, which is part of the `networkauth` distribution file.

```
PORTNAME=   networkauth
DISTVERSION=    ${QT5_VERSION}
```

```
USES=       qt-dist:5
```

If PORTNAME does not match the component name, it can be passed as an argument to qt-dist.

Пример 41. Building Qt 5 Components with Different Names

The port is Qt 5's gui component, which is part of the base distribution file.

```
PORTNAME=   gui
DISTVERSION=    ${QT5_VERSION}

USES=       qt-dist:5,base
```

15.78. readline

Possible arguments: (none), port

Uses readline as a library dependency, and sets CPPFLAGS and LDFLAGS as necessary. If the port argument is used or if readline is not present in the base system, add a dependency on devel/readline

15.79. samba

Possible arguments: build, env, lib, run

Handle dependency on Samba. env will not add any dependency and only set up the variables. build and run will add build-time and run-time dependency on smbd. lib will add a dependency on libsmbclient.so. The variables that are exported are:

SAMBAPORT
: The origin of the default Samba port.

SAMBAINCLUDES
: The location of the Samba header files.

SAMBALIBS
: The directory where the Samba shared libraries are available.

15.80. scons

Possible arguments: (none)

Provide support for the use of devel/scons. See Using scons for more information.

15.81. shared-mime-info

Possible arguments: (none)

Uses update-mime-database from misc/shared-mime-info. This uses will automatically add a post-install step in such a way that the port itself still can specify there own post-install step if needed. It also add an `@shared-mime-info` entry to the plist.

15.82. shebangfix

Possible arguments: (none)

A lot of software uses incorrect locations for script interpreters, most notably /usr/bin/perl and /bin/bash. The shebangfix macro fixes shebang lines in scripts listed in `SHEBANG_REGEX`, `SHEBANG_GLOB`, or `SHEBANG_FILES`.

`SHEBANG_REGEX`

Contains *one* extended regular expressions, and is used with the `-iregex` argument of find(1). See `USES=shebangfix` with `SHEBANG_REGEX`.

`SHEBANG_GLOB`

Contains a list of patterns used with the `-name` argument of find(1). See `USES=shebangfix` with `SHEBANG_GLOB`.

`SHEBANG_FILES`

Contains a list of files or sh(1) globs. The shebangfix macro is run from `${WRKSRC}`, so `SHEBANG_FILES` can contain paths that are relative to `${WRKSRC}`. It can also deal with absolute paths if files outside of `${WRKSRC}` require patching. See `USES=shebangfix` with `SHEBANG_FILES`.

Currently Bash, Java, Ksh, Lua, Perl, PHP, Python, Ruby, Tcl, and Tk are supported by default.

There are three configuration variables:

`SHEBANG_LANG`

The list of supported interpreters.

`_interp__CMD`

The path to the command interpreter on FreeBSD. The default value is `${LOCALBASE}/bin/interp`.

`_interp__OLD_CMD`

The list of wrong invocations of interpreters. These are typically obsolete paths, or paths used on other operating systems that are incorrect on FreeBSD. They will be replaced by the correct path in `_interp__CMD`.

These will *always* be part of `interp__OLD_CMD`: `"/usr/bin/env _interp"` `/bin/interp` `/usr/bin/interp` `/usr/local/bin/interp`.

`_interp__OLD_CMD` contain multiple values. Any entry with spaces must be

quoted. See Specifying all the Paths When Adding an Interpreter to USES=shebangfix.

The fixing of shebangs is done during the patch phase. If scripts are created with incorrect shebangs during the build phase, the build process (for example, the configure script, or the Makefiles) must be patched or given the right path (for example, with CONFIGURE_ENV, CONFIGURE_ARGS, MAKE_ENV, or MAKE_ARGS) to generate the right shebangs.

Correct paths for supported interpreters are available in _interp__CMD.

When used with USES=python, and the aim is only to fix the shebangs but a dependency on Python itself is not wanted, use PYTHON_NO_DEPENDS=yes.

Пример 42. Adding Another Interpreter to USES=shebangfix

To add another interpreter, set SHEBANG_LANG. For example:

```
SHEBANG_LANG=	lua
```

Пример 43. Specifying all the Paths When Adding an Interpreter to USES=shebangfix

If it was not already defined, and there were no default values for _interpOLD_CMD and _interpCMD the Ksh entry could be defined as:

```
SHEBANG_LANG=	ksh
ksh_OLD_CMD=	"/usr/bin/env ksh" /bin/ksh /usr/bin/ksh
ksh_CMD=	${LOCALBASE}/bin/ksh
```

Пример 44. Adding a Strange Location for an Interpreter

Some software uses strange locations for an interpreter. For example, an application might expect Python to be located in /opt/bin/python2.7. The strange path to be replaced can be declared in the port Makefile:

```
python_OLD_CMD= /opt/bin/python2.7
```

Пример 45. USES=shebangfix with SHEBANG_REGEX

To fix all the files in ${WRKSRC}/scripts ending in .pl, .sh, or .cgi do:

```
USES=	shebangfix
```

```
SHEBANG_REGEX=  ./scripts/.*\.(sh|pl|cgi)
```

SHEBANG_REGEX is used by running find -E, which uses modern regular expressions also known as extended regular expressions. See re_format(7) for more information.

Пример 46. USES=shebangfix *with* SHEBANG_GLOB

To fix all the files in ${WRKSRC} ending in .pl or .sh, do:

```
USES=   shebangfix
SHEBANG_GLOB=   *.sh *.pl
```

Пример 47. USES=shebangfix *with* SHEBANG_FILES

To fix the files script/foobar.pl and script/*.sh in ${WRKSRC}, do:

```
USES=   shebangfix
SHEBANG_FILES=  scripts/foobar.pl scripts/*.sh
```

15.83. sqlite

Possible arguments: (none), 2, 3

Add a dependency on SQLite. The default version used is 3, but version 2 is also possible using the :2 modifier.

15.84. ssl

Possible arguments: (none), build, run

Provide support for OpenSSL. A build- or run-time only dependency can be specified using build or run. These variables are available for the port's use, they are also added to MAKE_ENV:

OPENSSLBASE
: Path to the OpenSSL installation base.

OPENSSLDIR
: Path to OpenSSL's configuration files.

OPENSSLLIB
: Path to the OpenSSL libraries.

`OPENSSLINC`

Path to the OpenSSL includes.

`OPENSSLRPATH`

If defined, the path the linker needs to use to find the OpenSSL libraries.

If a port does not build with an OpenSSL flavor, set the `BROKEN_SSL` variable, and possibly the `BROKEN_SSL_REASON__flavor_`:

```
BROKEN_SSL= libressl
BROKEN_SSL_REASON_libressl= needs features only available in OpenSSL
```

15.85. `tar`

Possible arguments: (none), `Z`, `bz2`, `bzip2`, `lzma`, `tbz`, `tbz2`, `tgz`, `txz`, `xz`

Set `EXTRACT_SUFX` to `.tar`, `.tar.Z`, `.tar.bz2`, `.tar.bz2`, `.tar.lzma`, `.tbz`, `.tbz2`, `.tgz`, `.txz` or `.tar.xz` respectively.

15.86. `tcl`

Possible arguments: *version*, `wrapper`, `build`, `run`, `tea`

Add a dependency on Tcl. A specific version can be requested using *version*. The version can be empty, one or more exact version numbers (currently `84`, `85`, or `86`), or a minimal version number (currently `84+`, `85+` or `86+`). To only request a non version specific wrapper, use `wrapper`. A build- or run-time only dependency can be specified using `build` or `run`. To build the port using the Tcl Extension Architecture, use `tea`. After including bsd.port.pre.mk the port can inspect the results using these variables:

- `TCL_VER`: chosen major.minor version of Tcl
- `TCLSH`: full path of the Tcl interpreter
- `TCL_LIBDIR`: path of the Tcl libraries
- `TCL_INCLUDEDIR`: path of the Tcl C header files
- `TK_VER`: chosen major.minor version of Tk
- `WISH`: full path of the Tk interpreter
- `TK_LIBDIR`: path of the Tk libraries
- `TK_INCLUDEDIR`: path of the Tk C header files

15.87. `terminfo`

Possible arguments: (none)

Adds `@terminfo` to the plist. Use when the port installs *.terminfo files in ${PREFIX}/share/misc.

15.88. tk

Same as arguments for tcl

Small wrapper when using both Tcl and Tk. The same variables are returned as when using Tcl.

15.89. uidfix

Possible arguments: (none)

Changes some default behavior (mostly variables) of the build system to allow installing this port as a normal user. Try this in the port before using USES=fakeroot or patching.

15.90. uniquefiles

Possible arguments: (none), dirs

Make files or directories 'unique', by adding a prefix or suffix. If the dirs argument is used, the port needs a prefix (and only a prefix) based on UNIQUE_PREFIX for standard directories DOCSDIR, EXAMPLESDIR, DATADIR, WWWDIR, ETCDIR. These variables are available for ports:

- UNIQUE_PREFIX: The prefix to be used for directories and files. Default: ${PKGNAMEPREFIX}.
- UNIQUE_PREFIX_FILES: A list of files that need to be prefixed. Default: empty.
- UNIQUE_SUFFIX: The suffix to be used for files. Default: ${PKGNAMESUFFIX}.
- UNIQUE_SUFFIX_FILES: A list of files that need to be suffixed. Default: empty.

15.91. varnish

Possible arguments: 4, 6

Handle dependencies on Varnish Cache. 4 will add a dependency on www/varnish4. 6 will add a dependency on www/varnish6.

15.92. webplugin

Possible arguments: (none), ARGS

Automatically create and remove symbolic links for each application that supports the webplugin framework. ARGS can be one of:

- gecko: support plug-ins based on Gecko
- native: support plug-ins for Gecko, Opera, and WebKit-GTK
- linux: support Linux plug-ins
- all (default, implicit): support all plug-in types
- (individual entries): support only the browsers listed

These variables can be adjusted:

- `WEBPLUGIN_FILES`: No default, must be set manually. The plug-in files to install.
- `WEBPLUGIN_DIR`: The directory to install the plug-in files to, default PREFIX/lib/browser_plugins/WEBPLUGIN_NAME. Set this if the port installs plug-in files outside of the default directory to prevent broken symbolic links.
- `WEBPLUGIN_NAME`: The final directory to install the plug-in files into, default `PKGBASE`.

15.93. `xfce`

Possible arguments: (none), `gtk2`

Provide support for Xfce related ports. See Using Xfce for details.

The `gtk2` argument specifies that the port requires GTK2 support. It adds additional features provided by some core components, for example, x11/libxfce4menu and x11-wm/xfce4-panel.

15.94. `xorg`

Possible arguments: (none)

Provides an easy way to depend on X.org components. The components should be listed in `USE_XORG`. The available components are:

Таблица 49. Available X.Org Components

Name	Description
`dmx`	DMX extension library
`fontenc`	The fontenc Library
`fontutil`	Create an index of X font files in a directory
`ice`	Inter Client Exchange library for X11
`libfs`	The FS library
`pciaccess`	Generic PCI access library
`pixman`	Low-level pixel manipulation library
`sm`	Session Management library for X11
`x11`	X11 library
`xau`	Authentication Protocol library for X11
`xaw`	X Athena Widgets library
`xaw6`	X Athena Widgets library
`xaw7`	X Athena Widgets library
`xbitmaps`	X.Org bitmaps data
`xcb`	The X protocol C-language Binding (XCB) library

Name	Description
`xcomposite`	X Composite extension library
`xcursor`	X client-side cursor loading library
`xdamage`	X Damage extension library
`xdmcp`	X Display Manager Control Protocol library
`xext`	X11 Extension library
`xfixes`	X Fixes extension library
`xfont`	X font library
`xfont2`	X font library
`xft`	Client-sided font API for X applications
`xi`	X Input extension library
`xinerama`	X11 Xinerama library
`xkbfile`	XKB file library
`xmu`	X Miscellaneous Utilities libraries
`xmuu`	X Miscellaneous Utilities libraries
`xorg-macros`	X.Org development aclocal macros
`xorg-server`	X.Org X server and related programs
`xorgproto`	xorg protocol headers
`xpm`	X Pixmap library
`xpresent`	X Present Extension library
`xrandr`	X Resize and Rotate extension library
`xrender`	X Render extension library
`xres`	X Resource usage library
`xscrnsaver`	The XScrnSaver library
`xshmfence`	Shared memory 'SyncFence' synchronization primitive
`xt`	X Toolkit library
`xtrans`	Abstract network code for X
`xtst`	X Test extension
`xv`	X Video Extension library
`xvmc`	X Video Extension Motion Compensation library
`xxf86dga`	X DGA Extension
`xxf86vm`	X Vidmode Extension

15.95. `xorg-cat`

Possible arguments: `app`, `data`, `doc`, `driver`, `font`, `lib`, `proto`, `util`, `xserver` and (none) or one off `autotools` (default), `meson`

Provide support for building Xorg components. It takes care of setting up common dependencies and an appropriate configuration environment needed. This is intended only for Xorg components.

The category has to match upstream categories.

The second argument is the build system to use. autotools is the default, but meson is also supported.

15.96. `zip`

Possible arguments: (none), `infozip`

Indicates that the distribution files use the ZIP compression algorithm. For files using the InfoZip algorithm the `infozip` argument must be passed to set the appropriate dependencies.

Глава 16. Значения `__FreeBSD_version`

Here is a convenient list of `__FreeBSD_version` values as defined in sys/param.h:

16.1. FreeBSD 13 Versions

Таблица 50. FreeBSD 13 `__FreeBSD_version` Values

Value	Revision	Date	Release
1300000	339436	October 19, 2018	13.0-CURRENT.
1300001	339730	October 25, 2018	13.0-CURRENT after bumping OpenSSL shared library version numbers.
1300002	339765	October 25, 2018	13.0-CURRENT after restoration of sys/joystick.h.
1300003	340055	November 2, 2018	13.0-CURRENT after vop_symlink API change (`a_target` is now `const`.)
1300004	340841	November 23, 2018	13.0-CURRENT after enabling crtbegin and crtend code.
1300005	341836	December 11, 2018	13.0-CURRENT after enabling UFS inode checksums.
1300006	342398	December 24, 2018	13.0-CURRENT after fixing `sys/random.h` include to be usable from C++.
1300007	342629	December 30, 2018	13.0-CURRENT after changing the size of `struct linux_cdev` on 32-bit platforms.
1300008	342772	January 4, 2019	13.0-CURRENT after adding `kern.smp.threads_per_core` and `kern.smp.cores` sysctls.

Value	Revision	Date	Release
1300009	343213	January 20, 2019	13.0-CURRENT after `struct ieee80211vap` structure change to resolve ioctl/detach race for ieee80211com structure.
1300010	343485	January 27, 2019	13.0-CURRENT after increasing `SPECNAMELEN` from 63 to MAXNAMELEN (255).
1300011	344041	February 12, 2019	13.0-CURRENT after renameat(2) has been corrected to work with kernels built with the `CAPABILITIES` option.
1300012	344062	February 12, 2019	13.0-CURRENT after `taskqgroup_attach()` and `taskqgroup_attach_cpu()` take a device_t and a struct resource pointer as arguments for denoting device interrupts.
1300013	344300	February 19, 2019	13.0-CURRENT after the removal of drm and drm2.
1300014	344779	March 4, 2019	13.0-CURRENT after upgrading clang, llvm, lld, lldb, compiler-rt and libc++ to 8.0.0 rc3.
1300015	345196	March 15, 2019	13.0-CURRENT after deanonymizing thread and proc state enums, so userland applications can use them without redefining the value names.
1300016	345236	March 16, 2019	13.0-CURRENT after enabling LLVM OpenMP 8.0.0 rc5 on amd64 by default.

Value	Revision	Date	Release
1300017	345305	March 19, 2019	13.0-CURRENT after exposing the Rx mbuf buffer size to drivers in iflib.
1300018	346012	March 16, 2019	13.0-CURRENT after introduction of funlinkat syscall in 345982.
1300019	346282	April 16, 2019	13.0-CURRENT after addition of is_random_seeded(9) to random(4).
1300020	346358	April 18, 2019	13.0-CURRENT after restoring random(4) availability tradeoff prior to 346250 and adding new tunables and diagnostic sysctls for programmatically discovering early seeding problems after boot.
1300021	346645	April 24, 2019	13.0-CURRENT after LinuxKPI uses bus_dma(9) to be compatible with an IOMMU.
1300022	347089	May 4, 2019	13.0-CURRENT after fixing regression issue after 346645 in the LinuxKPI.
1300023	347192	May 6, 2019	13.0-CURRENT after list-ifying kernel dump device configuration.
1300024	347325	May 8, 2019	13.0-CURRENT after bumping the Mellanox driver version numbers (mlx4en(4); mlx5en(4)).
1300025	347532	May 13, 2019	13.0-CURRENT after renaming `vm.max_wired` to `vm.max_user_wired` and changing its type.

Value	Revision	Date	Release
1300026	347596	May 14, 2019	13.0-CURRENT after adding context member to ww_mutex in LinuxKPI.
1300027	347601	May 14, 2019	13.0-CURRENT after adding prepare to pm_ops in LinuxKPI.
1300028	347925	May 17, 2019	13.0-CURRENT after removal of bm, cs, de, ed, ep, ex, fe, pcn, sf, sn, tl, tx, txp, vx, wb, and xe drivers.
1300029	347984	May 20, 2019	13.0-CURRENT after removing some header pollution due to `sys/eventhandler.h`. Affected files may now need to explicitly include one or more of `sys/eventhandler.h`, `sys/ktr.h`, `sys/lock.h`, or `sys/mutex.h`, when the missing header may have been included implicitly prior to 1300029.
1300030	348350	May 29, 2019	13.0-CURRENT after adding relocation support to libdwarf on powerpc64 to fix handling of DWARF information on unlinked objects. Original commit in 348347.

Value	Revision	Date	Release
1300031	348808	June 8, 2019	13.0-CURRENT after adding dpcpu and vnet section fixes to i386 kernel modules to avoid panics in certain conditions. i386 kernel modules need to be recompiled with the linker script magic in place or they will refuse to load.
1300032	349151	June 17, 2019	13.0-CURRENT after separating kernel crc32() implementation to its own header (gsb_crc32.h) and renaming the source to gsb_crc32.c.
1300033	349277	June 21, 2019	13.0-CURRENT after additions to LinuxKPI's rcu list.
1300034	349352	June 24, 2019	13.0-CURRENT after NAND and NANDFS removal.
1300035	349846	July 8, 2019	13.0-CURRENT after merging the vm_page hold and wire mechanisms.
1300036	349972	July 13, 2019	13.0-CURRENT after adding arm_drain_writebuf() and arm_sync_icache() for compatibility with NetBSD and OpenBSD.
1300037	350307	July 24, 2019	13.0-CURRENT after removal of libcap_random(3).
1300038	350437	July 30, 2019	13.0-CURRENT after removal of gzip'ed a.out support.
1300039	350665	August 7, 2019	13.0-CURRENT after merge of fusefs from projects/fuse2.

Value	Revision	Date	Release
1300040	351140	August 16, 2019	13.0-CURRENT after deletion of sys/dir.h which has been deprecated since 1997.
(not changed)	351423	August 23, 2019	13.0-CURRENT after changing most arguments to ping6(8).
1300041	351480	August 25, 2019	13.0-CURRENT after removal of zlib 1.0.4 after the completion of kernel zlib unification.
1300042	351522	August 27, 2019	13.0-CURRENT after addition of kernel-side support for in-kernel TLS.
1300043	351698	September 2, 2019	13.0-CURRENT after removal of gets(3).
1300044	351701	September 2, 2019	13.0-CURRENT after adding sysfs create/remove functions that handles multiple files in one call to the LinuxKPI.
1300045	351729	September 3, 2019	13.0-CURRENT after adding sysctlbyname system call
1300046	351937	September 6, 2019	13.0-CURRENT after LinuxKPI sysfs improvements.
1300047	352110	September 9, 2019	13.0-CURRENT after changing the synchonization rules for vm_page reference counting..
1300048	352700	September 25, 2019	13.0-CURRENT after adding a shm_open2 syscall to support the upcoming memfd_create syscall.

Value	Revision	Date	Release
1300049	353274	October 7, 2019	13.0-CURRENT after factoring out the VNET shutdown check into an own vnet structure field.
1300050	353358	October 9, 2019	13.0-CURRENT after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 9.0.0 final release r372316.
1300051	353685	October 17, 2019	13.0-CURRENT after splitting out a more generic debugnet(4) from netdump(4).
1300052	353698	October 17, 2019	13.0-CURRENT after promoting the page busy field to a first class lock that no longer requires the object lock for consistency.
1300053	353700	October 17, 2019	13.0-CURRENT after implementing NetGDB.
1300054	353868	October 21, 2019	13.0-CURRENT after removing obsoleted KPIs that were used to access interface address lists.
1300055	354335	November 4, 2019	13.0-CURRENT after enabling device class group attributes in the LinuxKPI.
1300056	354460	November 7, 2019	13.0-CURRENT after fixing a potential OOB read security issue in libc++.
1300057	354694	November 13, 2019	13.0-CURRENT after adding support for AT_EXECPATH to elf_aux_info(3).

Value	Revision	Date	Release
1300058	354820	November 18, 2019	13.0-CURRENT after widening the vm_page aflags field to 16 bits.
1300059	354835	November 18, 2019	13.0-CURRENT after converting the in-tree sysent targets to use the new makesyscalls.lua.
1300060	354922	November 20, 2019	13.0-CURRENT after adding /etc/os-release as a symbolic link to /var/run/os-release.
1300061	354977	November 21, 2019	13.0-CURRENT after adding functions to bitstring(3) to find contiguous sequences of set or unset bits.
1300062	355309	December 2, 2019	13.0-CURRENT after adding TCP_STATS support.
1300063	355537	December 8, 2019	13.0-CURRENT after removal of VI_DOOMED (use VN_IS_DOOMED instead).
1300064	355658	December 9, 2019	13.0-CURRENT after correcting the C++ version check for declaring timespec_get(3).
1300065	355643	December 12, 2019	13.0-CURRENT after adding sigsetop extensions commonly found in musl libc and glibc.
1300066	355679	December 12, 2019	13.0-CURRENT after changing the internal interface between the NFS modules as part of the introduction of NFS 4.2.

Value	Revision	Date	Release
1300067	355732	December 13, 2019	13.0-CURRENT after removing the deprecated `callout_handle_init`, `timeout`, and `untimeout` functions.
1300068	355828	December 16, 2019	13.0-CURRENT after doubling the value of `ARG_MAX`, for 64 bit platforms.
1300069	356051	December 24, 2019	13.0-CURRENT after the addition of busdma templates.
1300070	356113	December 27, 2019	13.0-CURRENT after eliminating the last MI difference in AT_* definitions (for powerpc).
1300071	356135	December 27, 2019	13.0-CURRENT after making USB statistics be per-device instead of per bus.
1300072	356185	December 29, 2019	13.0-CURRENT after removal of GEOM_SCHED class and gsched tool.
1300073	356263	January 2, 2020	13.0-CURRENT after removing arm/arm as a valid target.
1300074	356337	January 3, 2020	13.0-CURRENT after removing flags argument from VOP_UNLOCK.
1300075	356409	January 6, 2020	13.0-CURRENT after adding own counter for cancelled USB transfers.
1300076	356511	January 8, 2020	13.0-CURRENT after pushing vnop implementation into the fileop layer in posix_fallocate.

Value	Revision	Date	Release
(not changed)	357396	February 2, 2020	13.0-CURRENT after removal of armv5 architecture code from the src tree.
1300077	357455	February 3, 2020	13.0-CURRENT after removal of sparc64 architecture code from the src tree.
1300078	358020	February 17, 2020	13.0-CURRENT after changing `struct vnet` and the VNET magic cookie.
1300079	358164	February 20, 2020	13.0-CURRENT after upgrading ncurses to 6.2.x
1300080	358172	February 20, 2020	13.0-CURRENT after adding realpathat syscall to VFS.
1300081	358218	February 21, 2020	13.0-CURRENT after after recent linuxkpi changes.
1300082	358497	March 1, 2020	13.0-CURRENT after removal of bktr(4).
1300083	358834	March 10, 2020	13.0-CURRENT after removal of amd(8), r358821.
1300084	358851	March 10, 2020	13.0-CURRENT after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.0-rc3 c290cb61fdc.
1300085	359261	March 23, 2020	13.0-CURRENT after the import of the kyua test framework.
1300086	359347	March 26, 2020	13.0-CURRENT after switching powerpc and powerpcspe to the lld linker.

Value	Revision	Date	Release
1300087	359374	March 27, 2020	13.0-CURRENT after refactoring the driver and consumer interfaces for in-kernel cryptography.
1300088	359530	April 1, 2020	13.0-CURRENT after removing support for procfs process debugging.
1300089	359727	April 8, 2020	13.0-CURRENT after cloning the RCU interface into a sleepable and a non-sleepable part in the LinuxKPI.
1300090	359747	April 9, 2020	13.0-CURRENT after removing the old NFS lock device driver that uses Giant.
1300091	359839	April 12, 2020	13.0-CURRENT after implementing a close_range(2) syscall.
1300092	359920	April 14, 2020	13.0-CURRENT after reworking unmapped mbufs in KTLS to carry ext_pgs in the mbuf itself.
1300093	360418	April 27, 2020	13.0-CURRENT after adding support for kernel TLS receive offload.
1300094	360796	May 7, 2020	13.0-CURRENT after linuxkpi changes.
1300095	361275	May 20, 2020	13.0-CURRENT after adding HyperV socket support for FreeBSD guests.

Value	Revision	Date	Release
1300096	361410	May 23, 2020	13.0-CURRENT after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.1 rc1 f79cd71e145.
1300097	361724	June 2, 2020	13.0-CURRENT after implementing __is_constexpr() function macro in the LinuxKPI.
1300098	362159	June 14, 2020	13.0-CURRENT after changing the `export_args ex_flags` field so that is 64bits.
1300099	362453	June 20, 2020	13.0-CURRENT after making liblzma use libmd implementation of SHA256.
1300100	362640	June 26, 2020	13.0-CURRENT after changing the internal API between the NFS kernel modules.
1300101	363077	July 10, 2020	13.0-CURRENT after implementing the array_size() function in the LinuxKPI.
1300102	363562	July 26, 2020	13.0-CURRENT after implementing lockless lookup in the VFS layer.
1300103	363757	August 1, 2020	13.0-CURRENT after making rights mandatory for NDINIT_ALL.
1300104	363783	August 2, 2020	13.0-CURRENT after vnode layout changes.
1300105	363894	August 5, 2020	13.0-CURRENT after vaccess() change.

Value	Revision	Date	Release
1300106	364092	August 11, 2020	13.0-CURRENT after adding an argument to newnfs_connect() that indicates use TLS for the connection.
1300107	364109	August 11, 2020	13.0-CURRENT after change to clone the task struct fields related to RCU.
1300108	364233	August 14, 2020	13.0-CURRENT after adding a few wait_bit functions to the linuxkpi, which are needed for DRM from Linux v5.4.
1300109	364274	August 16, 2020	13.0-CURRENT after vget() argument removal and namei flags renumbering.
(not changed)	364284	August 16, 2020	13.0-CURRENT after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to release/11.x llvmorg-11.0.0-rc1-47-gff47911ddfc.
1300110	364331	August 18, 2020	13.0-CURRENT after deleting the unused `use_ext` argument to `nfscl_reqstart()`.
1300111	364476	August 22, 2020	13.0-CURRENT after adding TLS support to the kernel RPC.
1300112	364747	August 25, 2020	13.0-CURRENT after merging OpenZFS support.
1300113	364753	August 25, 2020	13.0-CURRENT after adding atomic and bswap functions to libcompiler_rt.

Value	Revision	Date	Release
1300114	365459	September 8, 2020	13.0-CURRENT after changing arm64 AT_HWCAP definitions for elf_aux_info(3).
1300115	365705	September 14, 2020	13.0-CURRENT after fixing crunchgen(1) application build with `WARNS=6`.

16.2. FreeBSD 12 Versions

Таблица 51. FreeBSD 12 `__FreeBSD_version` Values

Value	Revision	Date	Release
1200000	302409	July 7, 2016	12.0-CURRENT.
1200001	302628	July 12, 2016	12.0-CURRENT after removing collation from `[a-z]`-type ranges.
1200002	304395	August 18, 2016	12.0-CURRENT after removing unused and obsolete `openbsd_poll` system call.
1200003	304608	August 22, 2016	12.0-CURRENT after adding C++11 `thread_local` support in rev 303795.
1200004	304752	August 24, 2016	12.0-CURRENT after fixing LC*MASK for newlocale(3) and querylocale(3) (rev 304703).
1200005	304789	August 25, 2016	12.0-CURRENT after changing some ioctl interfaces in rev 304787 between the iSCSI userspace programs and the kernel.
1200006	305256	September 1, 2016	12.0-CURRENT after crunchgen(1) META_MODE fix in 305254.

Value	Revision	Date	Release
1200007	305421	September 5, 2016	12.0-CURRENT after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).
1200008	305833	September 15, 2016	12.0-CURRENT after removing the 4.3BSD compatible macro `m_copy()` in 305824.
1200009	306077	September 21, 2016	12.0-CURRENT after removing `bio_taskqueue()` in 305988.
1200010	306276	September 23, 2016	12.0-CURRENT after mounting msdosfs(5) with longnames support by default.
1200011	306556	October 1, 2016	12.0-CURRENT after adding `fb_memattr` field to `fb_info` in 306555.
1200012	306592	October 2, 2016	12.0-CURRENT after net80211(4) changes (rev 306590, 306591).
1200013	307140	October 12, 2016	12.0-CURRENT after installing header files required development with libzfs_core.
1200014	307529	October 17, 2016	12.0-CURRENT after merging common code in rtwn(4) and urtwn(4), and adding support for 802.11ac devices.
1200015	308874	November 20, 2016	12.0-CURRENT after some ABI change for unbreaking powerpc.
1200016	309017	November 22, 2016	12.0-CURRENT after removing `PG_CACHED` -related fields from `vmmeter`.

Value	Revision	Date	Release
1200017	309124	November 25, 2016	12.0-CURRENT after upgrading our copies of clang, llvm, lldb, compiler-rt and libc++ to 3.9.0 release, and adding lld 3.9.0.
1200018	309676	December 7, 2016	12.0-CURRENT after adding the `ki_moretdname` member to `struct kinfo_proc` and `struct kinfo_proc32` to export the whole thread name to user-space utilities.
1200019	310149	December 16, 2016	12.0-CURRENT after starting to lay down the foundation for 11ac support.
1200020	312087	January 13, 2017	12.0-CURRENT after removing `fgetsock` and `fputsock`.
1200021	313858	February 16, 2017	12.0-CURRENT after removing MCA and EISA support.
1200022	314040	February 21, 2017	12.0-CURRENT after making the LinuxKPI task struct persistent across system calls.
(not changed)	314373	March 2, 2017	12.0-CURRENT after removing System V Release 4 binary compatibility support.
1200023	314564	March 2, 2017	12.0-CURRENT after upgrading our copies of clang, llvm, lld, lldb, compiler-rt and libc++ to 4.0.0.
1200024	314865	March 7, 2017	12.0-CURRENT after removal of pcap-int.h

Value	Revision	Date	Release
1200025	315430	March 16, 2017	12.0-CURRENT after addition of the <dev/mmc/mmc_ioctl.h> header.
1200026	315662	March 16, 2017	12.0-CURRENT after hiding `struct inpcb` and `struct tcpcb` from userland.
1200027	315673	March 21, 2017	12.0-CURRENT after making CAM SIM lock optional.
1200028	316683	April 10, 2017	12.0-CURRENT after renaming `smp_no_rendevous_barrier()` to `smp_no_rendezvous_barrier()` in 316648.
1200029	317176	April 19, 2017	12.0-CURRENT after the removal of `struct vmmeter` from `struct pcpu` from 317061.
1200030	317383	April 24, 2017	12.0-CURRENT after removing NATM support including en(4), fatm(4), hatm(4), and patm(4).
1200031	318736	May 23, 2017	12.0-CURRENT after types `ino_t`, `dev_t`, `nlink_t` were extended to 64bit and `struct dirent` changed layout (also known as ino64).
1200032	319664	June 8, 2017	12.0-CURRENT after removal of `groff`.
1200033	320043	June 17, 2017	12.0-CURRENT after the type of the `struct event` member `data` was increased to 64bit, and ext structure members added.

Value	Revision	Date	Release
1200034	320085	June 19, 2017	12.0-CURRENT after the NFS client and server were changed so that they actually use the 64bit `ino_t`.
1200035	320317	June 24, 2017	12.0-CURRENT after the `MAP_GUARD` mmap(2) flag was added.
1200036	320347	June 26, 2017	12.0-CURRENT after changing `time_t` to 64 bits on powerpc (32-bit version).
1200037	320545	July 1, 2017	12.0-CURRENT after the cleanup and inlining of `bus_dmamap*` functions (320528).
1200038	320879	July 10, 2017	12.0-CURRENT after MMC CAM committed. (320844).
1200039	321369	July 22, 2017	12.0-CURRENT after upgrade of copies of clang, llvm, lld, lldb, compiler-rt and libc++ to 5.0.0 (trunk r308421).
1200040	321688	July 29, 2017	12.0-CURRENT after adding NFS client forced dismount support `umount -N`.
1200041	322762	August 21, 2017	12.0-CURRENT after WRFSBASE instruction become operational on amd64.
1200042	322900	August 25, 2017	12.0-CURRENT after PLPMTUD counters were changed to use counter(9).
1200043	322989	August 28, 2017	12.0-CURRENT after dropping x86 CACHE_LINE_SIZE down to 64 bytes.

Value	Revision	Date	Release
1200044	323349	September 8, 2017	12.0-CURRENT after implementing poll_wait() in the LinuxKPI.
1200045	323706	September 18, 2017	12.0-CURRENT after adding shared memory support to LinuxKPI. (323703).
1200046	323910	September 22, 2017	12.0-CURRENT after adding support for 32-bit compatibility IOCTLs to LinuxKPI.
1200047	324053	September 26, 2017	12.0-CURRENT after removing M_HASHTYPE_RSS_UDP_IPV4_EX. (324052).
1200048	324227	October 2, 2017	12.0-CURRENT after hiding `struct socket` and `struct unpcb` from userland.
1200049	324281	October 4, 2017	12.0-CURRENT after adding the `value.u16` field to `struct diocgattr_arg`.
1200050	324342	October 5, 2017	12.0-CURRENT after adding the `armv7 MACHINE_ARCH`. (324340).
1200051	324455	October 9, 2017	12.0-CURRENT after removing libstand.a as a public interface. (324454).
1200052	325028	October 26, 2017	12.0-CURRENT after fixing `ptrace()` to always clear the correct thread event when resuming.
1200053	325506	November 7, 2017	12.0-CURRENT after changing `struct mbuf` layout to add optional hardware timestamps for receive packets.

Value	Revision	Date	Release
1200054	325852	November 15, 2017	12.0-CURRENT after changing the layout of `struct vmtotal` to allow for reporting large memory counters.
1200055	327740	January 9, 2018	12.0-CURRENT after adding `cpucontrol -e` support.
1200056	327952	January 14, 2018	12.0-CURRENT after upgrading clang, llvm, lld, lldb, compiler-rt and libc++ to 6.0.0 (branches/release_60 r321788).
1200057	329033	February 8, 2018	12.0-CURRENT after applying a clang 6.0.0 fix to make the wine ports build correctly.
1200058	329166	February 12, 2018	12.0-CURRENT after the lua loader was committed.
1200059	330299	March 2, 2018	12.0-CURRENT after removing the declaration of `union semun` unless `_WANT_SEMUN` is defined. Also the removal of `struct mymsg` and the renaming of kernel-only members of `struct semid_ds` and `struct msgid_ds`.
1200060	330384	March 4, 2018	12.0-CURRENT after upgrading clang, llvm, lld, lldb, compiler-rt and libc++ to 6.0.0 release.
1200061	332100	April 6, 2018	12.0-CURRENT after changing syslog(3) to emit RFC 5424 formatted messages.

Value	Revision	Date	Release
1200062	332423	April 12, 2018	12.0-CURRENT after changing the Netmap API.
1200063	333446	May 10, 2018	12.0-CURRENT after reworking CTL frontend and backend options to use nv(3), allow creating multiple ioctl frontend ports.
1200064	334074	May 22, 2018	12.0-CURRENT after changing the ifnet address and multicast address TAILQ to CK_STAILQ.
1200065	334290	May 28, 2018	12.0-CURRENT after changing dwatch(1) to allow '-E code' to override profile EVENT_DETAILS.
1200066	334466	June 1, 2018	12.0-CURRENT after removal of in-kernel pmc tables for Intel.
1200067	334892	June 9, 2018	12.0-CURRENT after adding DW_LANG constants to libdwarf.
1200068	334930	June 12, 2018	12.0-CURRENT after changing the interface between the NFS modules.
1200069	335237	June 15, 2018	12.0-CURRENT after changing `struct kerneldumpheader` to version 4 (similar to version 2 in 11-STABLE and previous).
1200070	335873	July 2, 2018	12.0-CURRENT after inlining atomic(9) in modules on amd64 and i386 requiring all modules of consumers to be rebuilt for these architectures.

Value	Revision	Date	Release
1200071	335930	July 4, 2018	12.0-CURRENT after changing the ABI and API of epoch(9) (335924) requiring modules of consumers to be rebuilt.
1200072	335979	July 5, 2018	12.0-CURRENT after changing the ABI and API of `struct xinpcb` and friends.
1200073	336313	July 15, 2018	12.0-CURRENT after changing the ABI and API of `struct if_shared_ctx` and `struct if_softc_ctx` requiring modules of iflib(9) consumers to be rebuilt.
1200074	336360	July 16, 2018	12.0-CURRENT after updating the configuration of libstdc++ to make use of C99 functions.
1200075	336538	July 19, 2018	12.0-CURRENT after zfsloader being folded into loader, and after adding ntpd:ntpd as uid:gid 123:123, and after removing arm big-endian support (MACHINE_ARCH=arm eb).
1200076	336914	July 30, 2018	12.0-CURRENT after KPI changes to timespecadd.
1200077	337576	August 10, 2018	12.0-CURRENT after timespec_get(3) was added to the system.
1200078	337863	August 15, 2018	12.0-CURRENT after exec.created hook for jails.

Value	Revision	Date	Release
1200079	338061	August 19, 2018	12.0-CURRENT after converting `arc4random` to using the Chacha20 algorithm and deprecating `arc4random_stir` and `arc4random_addrandom`.
1200080	338172	August 22, 2018	12.0-CURRENT after removing the drm drivers.
1200081	338182	August 21, 2018	12.0-CURRENT after KPI changes to NVMe.
1200082	338285	August 24, 2018	12.0-CURRENT after reverting the removal of the drm drivers.
1200083	338331	August 26, 2018	12.0-CURRENT after removing `arc4random_stir` and `arc4random_addrandom`.
1200084	338478	September 5, 2018	12.0-CURRENT after updating objcopy(1) to properly handle little-endian MIPS64 object files.
1200085	339270	October 19, 2018	12.0-STABLE after updating OpenSSL to version 1.1.1.
1200086	339732	October 25, 2018	12.0-STABLE after updating OpenSSL shared library version numbers.
1200500	340471	November 16, 2018	12-STABLE after releng/12.0 was branched.
1200501	342801	January 6, 2019	12-STABLE after merge of fixing linux_destroy_dev() behaviour when there are still files open from the destroying cdev.

Value	Revision	Date	Release
1200502	343126	January 17, 2019	12-STABLE after enabling sys/random.h #include from C++.
1200503	344152	Febrary 15, 2019	12-STABLE after merge of fixing renameat(2) for CAPABILITIES kernels.
1200504	345169	March 15, 2019	12-STABLE after merging CCM for the benefit of the ZoF port.
1200505	345327	March 20, 2019	12-STABLE after merging support for selectively disabling ZFS without disabling loader.
1200506	346168	April 12, 2019	12-STABLE after merging llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp 8.0.0 final release r356365.
1200507	346337	April 17, 2019	12-STABLE after MFC of iflib changes in 345303, 345658, and partially of 345305.
1200508	346784	April 27, 2019	12-STABLE after ether_gen_addr availability.
1200509	347790	May 16, 2019	12-STABLE after bumping the Mellanox driver version numbers (mlx4en(4); mlx5en(4)).
1200510	348036	May 21, 2019	12-STABLE after change to struct in linuxkpi from 348035.
1200511	348243	May 24, 2019	12-STABLE after MFC of 347843: adding group_leader member to struct task_struct to the LinuxKPI.

Value	Revision	Date	Release
1200512	348245	May 24, 2019	12-STABLE after adding context member to ww_mutex in LinuxKPI.
1200513	349763	July 5, 2019	12-STABLE after MFC of epoch(9) changes: 349763, 340404, 340415, 340417, 340419, 340420.
1200514	350083	July 17, 2019	12-STABLE after additions to LinuxKPI's rcu list.
1200515	350877	August 11, 2019	12-STABLE after MFC of 349891 (reorganize the SRCS lists as one file per line, and then alphabetize them) and 349972 (add arm_sync_icache() and arm_drain_writebuf() sysarch syscall wrappers).
1200516	351276	August 20, 2019	12-STABLE after MFC of various changes to iflib 351276.
1200517	352076	September 9, 2019	12-STABLE after adding sysfs create/remove functions that handles multiple files in one call to the LinuxKPI.
1200518	352114	September 10, 2019	12-STABLE after additional updates to LinuxKPI's sysfs.
1200519	352351	September 15, 2019	12-STABLE after MFC of the new fusefs driver.
1201000	352546	September 20, 2019	releng/12.1 branched from stable/12@r352480.
1201500	352547	September 20, 2019	12-STABLE after branching releng/12.1.
1201501	354598	November 10, 2019	12-STABLE after fixing a potential OOB read security issue in libc++.

Value	Revision	Date	Release
1201502	354613	November 11, 2019	12-STABLE after enabling device class group attributes in the LinuxKPI.
1201503	354928	November 21, 2019	12-STABLE after adding support for AT_EXECPATH to elf_aux_info(3).
1201504	355658	November 10, 2019	12-STABLE after correcting the C++ version check for declaring timespec_get(3).
1201505	355899	December 19, 2019	12-STABLE after adding sigsetop extensions commonly found in musl libc and glibc.
1201506	355968	December 21, 2019	12-STABLE after doubling the value of `ARG_MAX`, for 64 bit platforms.
1201507	356306	January 2, 2020	12-STABLE after adding functions to bitstring(3) to find contiguous sequences of set or unset bits.
1201508	356394	January 6, 2020	12-STABLE after making USB statistics be per-device instead of per bus.
1201509	356460	January 7, 2020	12-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 9.0.0 final release r372316.
1201510	356679	January 13, 2020	12-STABLE after adding own counter for cancelled USB transfers.

Value	Revision	Date	Release
1201511	357333	January 31, 2020	12-STABLE after adding /etc/os-release as a symbolic link to /var/run/os-release.
1201512	357612	February 6, 2020	12-STABLE after recent LinuxKPI changes.
1201513	359957	Apr 15, 2020	12-STABLE after cloning the RCU interface into a sleepable and a non-sleepable part in the LinuxKPI.
1201514	360525	May 1, 2020	12-STABLE after implementing full bus_dma(9) support in the LinuxKPI and pulling in all dependencies.
1201515	360545	May 1, 2020	12-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.0 release.
1201516	360620	May 4, 2020	12-STABLE after moving `id_mapped` to end of `bus_dma_impl` structure to preserve KPI.
1201517	361350	May 21, 2020	12-STABLE after renaming `vm.max_wired` to `vm.max_user_wired` and changing its type.
1201518	362319	June 18, 2020	12-STABLE after implementing __is_constexpr() function macro in the LinuxKPI.
1201519	362916	July 4, 2020	12-STABLE after making liblzma use libmd implementation of SHA256.

Value	Revision	Date	Release
1201520	363494	July 24, 2020	12-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.1 release.
1201521	363790	August 3, 2020	12-STABLE after implementing the array_size() function in the LinuxKPI.
1201522	363832	August 4, 2020	12-STABLE after adding sysctlbyname system call.
1201523	364390	August 19, 2020	12-STABLE after change to clone the task struct fields related to RCU.
1201524	365356	September 5, 2020	12-STABLE after splitting XDR off into a separate kernel module, to minimize ZFS dependencies.
1201525	365471	September 8, 2020	12-STABLE after adding atomic and bswap functions to libcompiler_rt.
1201526	365608	September 10, 2020	12-STABLE after updating net80211 and kernel privilege checking API changes.
1202000	365618	September 11, 2020	releng/12.2 branched from stable/12@r365618.
1202500	365619	September 11, 2020	12-STABLE after branching releng/12.2.
1202501	365661	September 12, 2020	12-STABLE after followup commits to libcompiler_rt.
1202502	365816	September 16, 2020	12-STABLE after fixing crunchgen(1) application build with `WARNS=6`.

16.3. FreeBSD 11 Versions

Таблица 52. FreeBSD 11 `__FreeBSD_version` Values

Value	Revision	Date	Release
1100000	256284	October 10, 2013	11.0-CURRENT.
1100001	256776	October 19, 2013	11.0-CURRENT after addition of support for "first boot" rc.d scripts, so ports can make use of this.
1100002	257696	November 5, 2013	11.0-CURRENT after dropping support for historic ioctls.
1100003	258284	November 17, 2013	11.0-CURRENT after iconv changes.
1100004	259424	December 15, 2013	11.0-CURRENT after the behavior change of `gss_pseudo_random` introduced in 259286.
1100005	260010	December 28, 2013	11.0-CURRENT after 259951 - Do not coalesce entries in vm_map_stack(9).
1100006	261246	January 28, 2014	11.0-CURRENT after upgrades of libelf and libdwarf.
1100007	261283	January 30, 2014	11.0-CURRENT after upgrade of libc++ to 3.4 release.
1100008	261881	February 14, 2014	11.0-CURRENT after libc++ 3.4 ABI compatibility fix.
1100009	261991	February 16, 2014	11.0-CURRENT after upgrade of llvm/clang to 3.4 release.
1100010	262630	February 28, 2014	11.0-CURRENT after upgrade of ncurses to 5.9 release (rev 262629).
1100011	263102	March 13, 2014	11.0-CURRENT after ABI change in struct if_data.

Value	Revision	Date	Release
1100012	263140	March 14, 2014	11.0-CURRENT after removal of Novell IPX protocol support.
1100013	263152	March 14, 2014	11.0-CURRENT after removal of AppleTalk protocol support.
1100014	263235	March 16, 2014	11.0-CURRENT after renaming <sys/capability.h> to <sys/capsicum.h> to avoid a clash with similarly named headers in other operating systems. A compatibility header is left in place to limit build breakage, but will be deprecated in due course.
1100015	263620	March 22, 2014	11.0-CURRENT after `cnt` rename to `vm_cnt`.
1100016	263660	March 23, 2014	11.0-CURRENT after addition of `armv6hf` `TARGET_ARCH`.
1100017	264121	April 4, 2014	11.0-CURRENT after GCC support for `__block` definition.
1100018	264212	April 6, 2014	11.0-CURRENT after support for UDP-Lite protocol (RFC 3828).
1100019	264289	April 8, 2014	11.0-CURRENT after FreeBSD-SA-14:06.openssl (rev 264265).
1100020	265215	May 1, 2014	11.0-CURRENT after removing lindev in favor of having /dev/full by default (rev 265212).

Value	Revision	Date	Release
1100021	266151	May 6, 2014	11.0-CURRENT after src.opts.mk changes, decoupling make.conf(5) from `buildworld` (rev 265419).
1100022	266904	May 30, 2014	11.0-CURRENT after changes to strcasecmp(3), moving strcasecmp_l(3) and strncasecmp_l(3) from <string.h> to <strings.h> for POSIX 2008 compliance (rev 266865).
1100023	267440	June 13, 2014	11.0-CURRENT after the CUSE library and kernel module have been attached to the build by default.
1100024	267992	June 27, 2014	11.0-CURRENT after sysctl(3) API change.
1100025	268066	June 30, 2014	11.0-CURRENT after regex(3) library update to add ">" and "<" delimiters.
1100026	268118	July 1, 2014	11.0-CURRENT after the internal interface between the NFS modules, including the krpc, was changed by (rev 268115).
1100027	268441	July 8, 2014	11.0-CURRENT after FreeBSD-SA-14:17.kmem (rev 268431).
1100028	268945	July 21, 2014	11.0-CURRENT after hdestroy(3) compliance fix changed ABI.
1100029	270173	August 3, 2014	11.0-CURRENT after `SOCK_DGRAM` bug fix (rev 269489).

Value	Revision	Date	Release
1100030	270929	September 1, 2014	11.0-CURRENT after `SOCK_RAW` sockets were changed to not modify packets at all.
1100031	271341	September 9, 2014	11.0-CURRENT after FreeBSD-SA-14:18.openssl (rev 269686).
1100032	271438	September 11, 2014	11.0-CURRENT after API changes to `ifa_ifwithbroadaddr`, `ifa_ifwithdstaddr`, `ifa_ifwithnet`, and `ifa_ifwithroute`.
1100033	271657	September 9, 2014	11.0-CURRENT after changing `access`, `eaccess`, and `faccessat` to validate the mode argument.
1100034	271686	September 16, 2014	11.0-CURRENT after FreeBSD-SA-14:19.tcp (rev 271666).
1100035	271705	September 17, 2014	11.0-CURRENT after i915 HW context support.
1100036	271724	September 17, 2014	Version bump to have ABI note distinguish binaries ready for strict mmap(2) flags checking (rev 271724).
1100037	272674	October 6, 2014	11.0-CURRENT after addition of explicit_bzero(3) (rev 272673).
1100038	272951	October 11, 2014	11.0-CURRENT after cleanup of TCP wrapper headers.
1100039	273250	October 18, 2014	11.0-CURRENT after removal of `MAP_RENAME` and `MAP_NORESERVE`.

Value	Revision	Date	Release
1100040	273432	October 21, 2014	11.0-CURRENT after FreeBSD-SA-14:23 (rev 273146).
1100041	273875	October 30, 2014	11.0-CURRENT after API changes to `syscall_register`, `syscall32_register`, `syscall_register_helper` and `syscall32_register_helper` (rev 273707).
1100042	274046	November 3, 2014	11.0-CURRENT after a change to `struct tcpcb`.
1100043	274085	November 4, 2014	11.0-CURRENT after enabling vt(4) by default.
1100044	274116	November 4, 2014	11.0-CURRENT after adding new libraries/utilities (dpv and figpar) for data throughput visualization.
1100045	274162	November 4, 2014	11.0-CURRENT after FreeBSD-SA-14:23, FreeBSD-SA-14:24, and FreeBSD-SA-14:25.
1100046	274470	November 13, 2014	11.0-CURRENT after `kern_poll` signature change (rev 274462).
1100047	274476	November 13, 2014	11.0-CURRENT after removal of no-at version of VFS syscalls helpers, like `kern_open`.
1100048	275358	December 1, 2014	11.0-CURRENT after starting the process of removing the use of the deprecated "M_FLOWID" flag from the network code.

Value	Revision	Date	Release
1100049	275633	December 9, 2014	11.0-CURRENT after importing an important fix to the LLVM vectorizer, which could lead to buffer overruns in some cases.
1100050	275732	December 12, 2014	11.0-CURRENT after adding AES-ICM and AES-GCM to OpenCrypto.
1100051	276096	December 23, 2014	11.0-CURRENT after removing old NFS client and server code from the kernel.
1100052	276479	December 31, 2014	11.0-CURRENT after upgrade of clang, llvm and lldb to 3.5.0 release.
1100053	276781	January 7, 2015	11.0-CURRENT after MCLGET(9) gained a return value (rev 276750).
1100054	277213	January 15, 2015	11.0-CURRENT after rewrite of callout subsystem.
1100055	277528	January 22, 2015	11.0-CURRENT after reverting callout changes in 277213.
1100056	277610	January 23, 2015	11.0-CURRENT after addition of `futimens` and `utimensat` system calls.
1100057	277897	January 29, 2015	11.0-CURRENT after removal of d_thread_t.
1100058	278228	February 5, 2015	11.0-CURRENT after addition of support for probing the SCSI VPD Extended Inquiry page (0x86).

Value	Revision	Date	Release
1100059	278442	February 9, 2015	11.0-CURRENT after import of xz 5.2.0, which added multi-threaded compression and lzma gained libthr dependency (rev 278433).
1100060	278846	February 16, 2015	11.0-CURRENT after forwarding `FBIO_BLANK` to framebuffer clients.
1100061	278964	February 18, 2015	11.0-CURRENT after `CDAI_FLAG_NONE` addition.
1100062	279221	February 23, 2015	11.0-CURRENT after mtio(4) and sa(4) API and ioctl(2) additions.
1100063	279728	March 7, 2015	11.0-CURRENT after adding mutex support to the `pps_ioctl()` API in the kernel.
1100064	279729	March 7, 2015	11.0-CURRENT after adding PPS support to USB serial drivers.
1100065	280031	March 15, 2015	11.0-CURRENT after upgrading clang, llvm and lldb to 3.6.0.
1100066	280306	March 20, 2015	11.0-CURRENT after removal of SSLv2 support from OpenSSL.
1100067	280630	March 25, 2015	11.0-CURRENT after removal of SSLv2 support from fetch(1) and fetch(3).
1100068	281172	April 6, 2015	11.0-CURRENT after change to net.inet6.ip6.mif6table sysctl.
1100069	281550	April 15, 2015	11.0-CURRENT after removal of const qualifier from iconv(3).

Value	Revision	Date	Release
1100070	281613	April 16, 2015	11.0-CURRENT after moving ALTQ from contrib to net/altq.
1100071	282256	April 29, 2015	11.0-CURRENT after API/ABI change to smb(4) (rev 281985).
1100072	282319	May 1, 2015	11.0-CURRENT after adding reallocarray(3) in libc (rev 282314).
1100073	282650	May 8, 2015	11.0-CURRENT after extending the maximum number of allowed PCM channels in a PCM stream to 127 and decreasing the maximum number of sub-channels to 1.
1100074	283526	May 25, 2015	11.0-CURRENT after adding preliminary support for x86-64 Linux binaries (rev 283424), and upgrading clang and llvm to 3.6.1.
1100075	283623	May 27, 2015	11.0-CURRENT after `dounmount()` requiring a reference on the passed struct mount (rev 283602).
1100076	283983	June 4, 2015	11.0-CURRENT after disabled generation of legacy formatted password databases entries by default.
1100077	284233	June 10, 2015	11.0-CURRENT after API changes to `lim_cur`, `lim_max`, and `lim_rlimit` (rev 284215).
1100078	286672	August 12, 2015	11.0-CURRENT after crunchgen(1) changes from 284356 to 285986.

Value	Revision	Date	Release
1100079	286874	August 18, 2015	11.0-CURRENT after import of jemalloc 4.0.0 (rev 286866).
1100080	288943	October 5, 2015	11.0-CURRENT after upgrading clang, llvm, lldb, compiler-rt and libc++ to 3.7.0.
1100081	289415	October 16, 2015	11.0-CURRENT after undating ZFS to support resumable send/receive (rev 289362).
1100082	289594	October 19, 2015	11.0-CURRENT after Linux KPI updates.
1100083	289749	October 22, 2015	11.0-CURRENT after renaming linuxapi.ko to linuxkpi.ko.
1100084	290135	October 29, 2015	11.0-CURRENT after moving the LinuxKPI module into the default kernel build.
1100085	290207	October 30, 2015	11.0-CURRENT after import of OpenSSL 1.0.2d.
1100086	290275	November 2, 2015	11.0-CURRENT after making figpar(3) macros more unique.
1100087	290479	November 7, 2015	11.0-CURRENT after changing sysctl_add_oid(9)'s ABI.
1100088	290495	November 7, 2015	11.0-CURRENT after string collation and locales rework.
1100089	290505	November 7, 2015	11.0-CURRENT after API change to sysctl_add_oid(9) (rev 290475).
1100090	290715	November 10, 2015	11.0-CURRENT after API change to callout_stop macro; (rev 290664).

Value	Revision	Date	Release
1100091	291537	November 30, 2015	11.0-CURRENT after changing the interface between the nfsd.ko and nfscommon.ko modules in 291527.
1100092	292499	December 19, 2015	11.0-CURRENT after removal of vm_pageout_grow_cache (rev 292469).
1100093	292966	December 30, 2015	11.0-CURRENT after removal of sys/crypto/sha2.h (rev 292782).
1100094	294086	January 15, 2016	11.0-CURRENT after LinuxKPI PCI changes (rev 294086).
1100095	294327	January 19, 2016	11.0-CURRENT after LRO optimizations.
1100096	294505	January 21, 2016	11.0-CURRENT after LinuxKPI idr_* additions.
1100097	294860	January 26, 2016	11.0-CURRENT after API change to dpv(3).
1100098	295682	February 16, 2016	11.0-CURRENT after API change to rman (rev 294883).
1100099	295739	February 18, 2016	11.0-CURRENT after allowing drivers to set the TCP ACK/data segment aggregation limit.
1100100	296136	February 26, 2016	11.0-CURRENT after bus_alloc_resource_any(9) API addition.
1100101	296417	March 5, 2016	11.0-CURRENT after upgrading our copies of clang, llvm, lldb and compiler-rt to 3.8.0 release.
1100102	296749	March 12, 2016	11.0-CURRENT after libelf cross-endian fix in rev 296685.

Value	Revision	Date	Release
1100103	297000	March 18, 2016	11.0-CURRENT after using uintmax_t for rman ranges.
1100104	297156	March 21, 2016	11.0-CURRENT after tracking filemon usage via a proc.p_filemon pointer rather than its own lists.
1100105	297602	April 6, 2016	11.0-CURRENT after fixing sed functions `i` and `a` from discarding leading white space.
1100106	298486	April 22, 2016	11.0-CURRENT after fixes for using IPv6 addresses with RDMA.
1100107	299090	May 4, 2016	11.0-CURRENT after improving performance and functionality of the bitstring(3) api.
1100108	299530	May 12, 2016	11.0-CURRENT after fixing handling of IOCTLs in the LinuxKPI.
1100109	299933	May 16, 2016	11.0-CURRENT after implementing more Linux device related functions in the LinuxKPI.
1100110	300207	May 19, 2016	11.0-CURRENT after adding support for managing Shingled Magnetic Recording (SMR) drives.
1100111	300303	May 20, 2016	11.0-CURRENT after removing brk and sbrk from arm64.
1100112	300539	May 23, 2016	11.0-CURRENT after adding bit_count to the bitstring(3) API.
1100113	300701	May 26, 2016	11.0-CURRENT after disabling alignment faults on armv6.

Value	Revision	Date	Release
1100114	300806	May 26, 2016	11.0-CURRENT after fixing crunchgen(1) usage with `MAKEOBJDIRPREFIX`.
1100115	300982	May 30, 2016	11.0-CURRENT after adding an mbuf flag for `M_HASHTYPE_`.
1100116	301011	May 31, 2016	11.0-CURRENT after SHA-512t256 (rev 300903) and Skein (rev 300966) where added to libmd, libcrypt, the kernel, and ZFS (rev 301010).
1100117	301892	June 6, 2016	11.0-CURRENT after libpam was synced with stock 301602, bumping library version.
1100118	302071	June 21, 2016	11.0-CURRENT after breaking binary compatibility of struct disk 302069.
1100119	302150	June 23, 2016	11.0-CURRENT after switching geom_disk to using a pool mutex.
1100120	302153	June 23, 2016	11.0-CURRENT after adding spares to struct ifnet.
1100121	303979	August 12, 2015	11-STABLE after `releng/11.0` branched from 11-STABLE (rev 303975).
1100500	303979	August 12, 2016	11.0-STABLE adding branched 303976.
1100501	304609	August 22, 2016	11.0-STABLE after adding C++11 thread_local support.
1100502	304865	August 26, 2016	11.0-STABLE after `LC_*_MASK` fix.

Value	Revision	Date	Release
1100503	305733	September 12, 2016	11.0-STABLE after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).
1100504	307330	October 14, 2016	11.0-STABLE after ZFS merges.
1100505	307590	October 19, 2016	11.0-STABLE after `struct fb_info` change.
1100506	308048	October 28, 2016	11.0-STABLE after installing header files required development with libzfs_core.
1100507	310120	December 15, 2016	11.0-STABLE after adding the `ki_moretdname` member to `struct kinfo_proc` and `struct kinfo_proc32` to export the whole thread name to user-space utilities.
1100508	310618	December 26, 2016	11.0-STABLE after upgrading our copies of clang, llvm, lldb, compiler-rt and libc++ to 3.9.1 release, and adding lld 3.9.1.
1100509	311186	January 3, 2017	11.0-STABLE after crunchgen(1) META_MODE fix (rev 311185).
1100510	315312	March 15, 2017	11.0-STABLE after MFC of `fget_cap`, `getsock_cap`, and related changes.
1100511	316423	April 2, 2017	11.0-STABLE after multiple MFCs updating clang, llvm, lld, lldb, compiler-rt and libc++ to 4.0.0 release.

Value	Revision	Date	Release
1100512	316498	April 4, 2017	11.0-STABLE after making CAM SIM lock optional (revs 315673, 315674).
1100513	318197	May 11, 2017	11.0-STABLE after merging the addition of the <dev/mmc/mmc_ioctl.h> header.
1100514	319279	May 31, 2017	11.0-STABLE after multiple MFCs of `libpcap`, `WITHOUT_INET6`, and a few other minor changes.
1101000	320486	June 30, 2017	`releng/11.1` branched from `stable/11`.
1101001	320763	June 30, 2017	11.1-RC1 After merging the `MAP_GUARD` mmap(2) flag addition.
1101500	320487	June 30, 2017	11-STABLE after `releng/11.1` branched.
1101501	320666	July 5, 2017	11-STABLE after merging the `MAP_GUARD` mmap(2) flag addition.
1101502	321688	July 29, 2017	11-STABLE after merging the NFS client forced dismount support `umount -N` addition.
1101503	323431	September 11, 2017	11-STABLE after merging changes making the WRFSBASE instruction operational on amd64.
1101504	324006	September 26, 2017	11-STABLE after merging libm from head, which adds cacoshl(3), cacosl(3), casinhl(3), casinl(3), catanl(3), catanhl(3), sincos(3), sincosf(3), and sincosl(3).

Value	Revision	Date	Release
1101505	324023	September 26, 2017	11-STABLE after merging clang, llvm, lld, lldb, compiler-rt and libc++ 5.0.0 release.
1101506	325003	October 25, 2017	11-STABLE after merging 324281, adding the `value.u16` field to `struct diocgattr_arg`.
1101507	328379	January 24, 2018	11-STABLE after merging 325028, fixing `ptrace()` to always clear the correct thread event when resuming.
1101508	328386	January 24, 2018	11-STABLE after merging 316648, renaming smp_no_rendevous_barrier() to smp_no_rendezvous_barrier().
1101509	328653	February 1, 2018	11-STABLE after an overwrite merge backport of the LinuxKPI from FreeBSD-head.
1101510	329450	February 17, 2018	11-STABLE after the cmpxchg() macro is now fully functional in the LinuxKPI.
1101511	329981	February 25, 2018	11-STABLE after concluding the recent LinuxKPI related updates.
1101512	331219	March 19, 2018	11-STABLE after merging retpoline support from the upstream llvm, clang and lld 5.0 branches.

Value	Revision	Date	Release
1101513	331838	March 31, 2018	11-STABLE after merging clang, llvm, lld, lldb, compiler-rt and libc++ 6.0.0 release, and several follow-up fixes.
1101514	332089	April 5, 2018	11-STABLE after merging 328331, adding a new and incompatible interpretation of ${name}_limits in rc scripts.
1101515	332363	April 10, 2018	11-STABLE after reverting 331880, removing the new and incompatible interpretation of ${name}_limits in rc scripts.
1101516	334392	May 30, 2018	11-STABLE after dwatch(1) touch-ups.
1102000	334459	June 1, 2018	`releng/11.2` branched from `stable/11`.
1102500	334461	June 1, 2018	11-STABLE after releng/11.2 branched.
1102501	335436	June 20, 2018	11-STABLE after LinuxKPI updates requiring recompilation of external kernel modules.
1102502	338617	September 12, 2018	11-STABLE after adding a socket option SO_TS_CLOCK and fixing recvmsg32() system call to properly down-convert layout of the 64-bit structures to match what 32-bit app(s) expect.

Value	Revision	Date	Release
1102503	338931	September 25, 2018	11-STABLE after merging a TCP checksum fix to iflib(9) and adding new media types to if_media.h
1102504	340309	November 9, 2018	11-STABLE after several MFCs: updating objcopy(1) to properly handle little-endian MIPS64 object; correcting mips64el test to use ELF header; adding test for 64-bit ELF in _libelf_is_mips64el.
1102505	342804	January 6, 2019	11-STABLE after merge of fixing linux_destroy_dev() behaviour when there are still files open from the destroying cdev.
1102506	344220	February 17, 2019	11-STABLE after merging multiple commits to lualoader.
1102507	346296	April 16, 2019	11-STABLE after merging llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp 8.0.0 final release r356365.
1102508	346784	April 27, 2019	11-STABLE after ether_gen_addr availability.
1102509	347212	May 6, 2019	11-STABLE after MFC of 345303, 345658, and partially of 345305.
1102510	347883	May 16, 2019	11-STABLE after bumping the Mellanox driver version numbers (mlx4en(4); mlx5en(4)).
1103000	349026	June 14, 2019	`releng/11.3` branched from `stable/11`.

Value	Revision	Date	Release
1103500	349027	June 14, 2019	11-STABLE after releng/11.3 branched.
1103501	354598	November 10, 2019	11-STABLE after fixing a potential OOB read security issue in libc++.
1103502	354614	November 11, 2019	11-STABLE after adding sysfs create/remove functions that handles multiple files in one call to the LinuxKPI.
1103503	354615	November 11, 2019	11-STABLE after LinuxKPI sysfs improvements.
1103504	354616	November 11, 2019	11-STABLE after enabling device class group attributes in the LinuxKPI.
1103505	355899	December 19, 2019	11-STABLE after adding sigsetop extensions commonly found in musl libc and glibc.
1103506	356395	January 6, 2020	11-STABLE after making USB statistics be per-device instead of per bus.
1103507	356680	January 13, 2020	11-STABLE after adding own counter for cancelled USB transfers.
1103508	357613	February 6, 2020	11-STABLE after recent LinuxKPI changes.
1103509	359958	April 15, 2020	11-STABLE after moving `id_mapped` to end of `bus_dma_impl` structure to preserve KPI.
1103510	360658	May 5, 2020	11-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 9.0.0 final release r372316.

Value	Revision	Date	Release
1103511	360784	May 7, 2020	11-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.0 release.
1104000	360804	May 8, 2020	`releng/11.4` branched from `stable/11`.
1104001	360822	May 8, 2020	11.4-BETA1 after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.0 release.
1104500	360805	May 8, 2020	11-STABLE after releng/11.4 branched.
1104501	362320	June 18, 2020	11-STABLE after implementing __is_constexpr() function macro in the LinuxKPI.
1104502	362919	July 4, 2020	11-STABLE after making liblzma use libmd implementation of SHA256.
1104503	363496	July 24, 2020	11-STABLE after updating llvm, clang, compiler-rt, libc++, libunwind, lld, lldb and openmp to 10.0.1 release.
1104504	363792	August 3, 2020	11-STABLE after implementing the array_size() function in the LinuxKPI.
1104505	364391	August 19, 2020	11-STABLE after change to clone the task struct fields related to RCU.
1104506	365471	September 8, 2020	11-STABLE after adding atomic and bswap functions to libcompiler_rt.

Value	Revision	Date	Release
1104507	365661	September 12, 2020	11-STABLE after followup commits to libcompiler_rt.

16.4. FreeBSD 10 Versions

Таблица 53. FreeBSD 10 `__FreeBSD_version` Values

Value	Revision	Date	Release
1000000	225757	September 26, 2011	10.0-CURRENT.
1000001	227070	November 4, 2011	10-CURRENT after addition of the posix_fadvise(2) system call.
1000002	228444	December 12, 2011	10-CURRENT after defining boolean true/false in sys/types.h, sizeof(bool) may have changed (rev 228444). 10-CURRENT after xlocale.h was introduced (rev 227753).
1000003	228571	December 16, 2011	10-CURRENT after major changes to carp(4), changing size of struct in_aliasreq, struct in6_aliasreq (rev 228571) and straitening arguments check of SIOCAIFADDR (rev 228574).
1000004	229204	January 1, 2012	10-CURRENT after the removal of `skpc()` and the addition of memcchr(9) (rev 229200).
1000005	230207	January 16, 2012	10-CURRENT after the removal of support for SIOCSIFADDR, SIOCSIFNETMASK, SIOCSIFBRDADDR, SIOCSIFDSTADDR ioctls.

Value	Revision	Date	Release
1000006	230590	January 26, 2012	10-CURRENT after introduction of read capacity data asynchronous notification in the cam(4) layer.
1000007	231025	February 5, 2012	10-CURRENT after introduction of new tcp(4) socket options: TCP_KEEPINIT, TCP_KEEPIDLE, TCP_KEEPINTVL, and TCP_KEEPCNT.
1000008	231505	February 11, 2012	10-CURRENT after introduction of the new extensible sysctl(3) interface NET_RT_IFLISTL to query address lists.
1000009	232154	February 25, 2012	10-CURRENT after import of libarchive 3.0.3 (rev 232153).
1000010	233757	March 31, 2012	10-CURRENT after xlocale cleanup.
1000011	234355	April 16, 2012	10-CURRENT import of LLVM/Clang 3.1 trunk 154661 (rev 234353).
1000012	234924	May 2, 2012	10-CURRENT jemalloc import.
1000013	235788	May 22, 2012	10-CURRENT after byacc import.
1000014	237631	June 27, 2012	10-CURRENT after BSD sort becoming the default sort (rev 237629).
1000015	238405	July 12, 2012	10-CURRENT after import of OpenSSL 1.0.1c.
(not changed)	238429	July 13, 2012	10-CURRENT after the fix for LLVM/Clang 3.1 regression.

Value	Revision	Date	Release
1000016	239179	August 8, 2012	10-CURRENT after KBI change in ucom(4).
1000017	239214	August 8, 2012	10-CURRENT after adding streams feature to the USB stack.
1000018	240233	September 8, 2012	10-CURRENT after major rewrite of pf(4).
1000019	241245	October 6, 2012	10-CURRENT after pfil(9) KBI/KPI changed to supply packets in net byte order to AF_INET filter hooks.
1000020	241610	October 16, 2012	10-CURRENT after the network interface cloning KPI changed and struct if_clone becoming opaque.
1000021	241897	October 22, 2012	10-CURRENT after removal of support for non-MPSAFE filesystems and addition of support for FUSEFS (rev 241519).
1000022	241913	October 22, 2012	10-CURRENT after the entire IPv4 stack switched to network byte order for IP packet header storage.
1000023	242619	November 5, 2012	10-CURRENT after jitter buffer in the common USB serial driver code, to temporarily store characters if the TTY buffer is full. Add flow stop and start signals when this happens.
1000024	242624	November 5, 2012	10-CURRENT after clang was made the default compiler on i386 and amd64.

Value	Revision	Date	Release
1000025	243443	November 17, 2012	10-CURRENT after the sin6_scope_id member variable in struct sockaddr_in6 was changed to being filled by the kernel before passing the structure to the userland via sysctl or routing socket. This means the KAME-specific embedded scope id in sin6_addr.s6_addr[2] is always cleared in userland application.
1000026	245313	January 11, 2013	10-CURRENT after install gained the -N flag. May also be used to indicate the presence of nmtree.
1000027	246084	January 29, 2013	10-CURRENT after cat gained the -l flag (rev 246083).
1000028	246759	February 13, 2013	10-CURRENT after USB moved to the driver structure requiring a rebuild of all USB modules.
1000029	247821	March 4, 2013	10-CURRENT after the introduction of tickless callout facility which also changed the layout of struct callout (rev 247777).
1000030	248210	March 12, 2013	10-CURRENT after KPI breakage introduced in the VM subsystem to support read/write locking (rev 248084).

Value	Revision	Date	Release
1000031	249943	April 26, 2013	10-CURRENT after the dst parameter of the ifnet `if_output` method was changed to take const qualifier (rev 249925).
1000032	250163	May 1, 2013	10-CURRENT after the introduction of the accept4(2) (rev 250154) and pipe2(2) (rev 250159) system calls.
1000033	250881	May 21, 2013	10-CURRENT after flex 2.5.37 import.
1000034	251294	June 3, 2013	10-CURRENT after the addition of these functions to libm: cacos(3), cacosf(3), cacosh(3), cacoshf(3), casin(3), casinf(3), casinh(3), casinhf(3), catan(3), catanf(3), catanh(3), catanhf(3), logl(3), log2l(3), log10l(3), log1pl(3), expm1l(3).
1000035	251527	June 8, 2013	10-CURRENT after the introduction of the aio_mlock(2) system call (rev 251526).
1000036	253049	July 9, 2013	10-CURRENT after the addition of a new function to the kernel GSSAPI module's function call interface.

Value	Revision	Date	Release
1000037	253089	July 9, 2013	10-CURRENT after the migration of statistics structures to PCPU counters. Changed structures include: `ahstat`, `arpstat`, `espstat`, `icmp6_ifstat`, `icmp6stat`, `in6_ifstat`, `ip6stat`, `ipcompstat`, `ipipstat`, `ipsecstat`, `mrt6stat`, `mrtstat`, `pfkeystat`, `pim6stat`, `pimstat`, `rip6stat`, `udpstat` (rev 253081).
1000038	253396	July 16, 2013	10-CURRENT after making `ARM EABI` the default ABI on arm, armeb, armv6, and armv6eb architectures.
1000039	253549	July 22, 2013	10-CURRENT after `CAM` and mps(4) driver scanning changes.
1000040	253638	July 24, 2013	10-CURRENT after addition of libusb pkgconf files.
1000041	253970	August 5, 2013	10-CURRENT after change from `time_second` to `time_uptime` in `PF_INET6`.
1000042	254138	August 9, 2013	10-CURRENT after VM subsystem change to unify soft and hard busy mechanisms.
1000043	254273	August 13, 2013	10-CURRENT after `WITH_ICONV` is enabled by default. A new src.conf(5) option, `WITH_LIBICONV_COMPAT` (disabled by default) adds `libiconv_open` to provide compatibility with the libiconv port.

Value	Revision	Date	Release
1000044	254358	August 15, 2013	10-CURRENT after `libc.so` conversion to an ld(1) script (rev 251668).
1000045	254389	August 15, 2013	10-CURRENT after devfs programming interface change by replacing the cdevsw flag `D_UNMAPPED_IO` with the struct cdev flag `SI_UNMAPPED`.
1000046	254537	August 19, 2013	10-CURRENT after addition of `M_PROTO[9-12]` and removal of `M_FRAG|M_FIRSTFRAG|M_LASTFRAG` mbuf flags (rev 254524, 254526).
1000047	254627	August 21, 2013	10-CURRENT after stat(2) update to allow storing some Windows/DOS and CIFS file attributes as stat(2) flags.
1000048	254672	August 22, 2013	10-CURRENT after modification of structure `xsctp_inpcb`.
1000049	254760	August 24, 2013	10-CURRENT after physio(9) support for devices that do not function properly with split I/O, such as sa(4).
1000050	254844	August 24, 2013	10-CURRENT after modifications of structure `mbuf` (rev 254780, 254799, 254804, 254807254842).
1000051	254887	August 25, 2013	10-CURRENT after Radeon KMS driver import (rev 254885).
1000052	255180	September 3, 2013	10-CURRENT after import of NetBSD `libexecinfo` is connected to the build.

Value	Revision	Date	Release
1000053	255305	September 6, 2013	10-CURRENT after API and ABI changes to the Capsicum framework.
1000054	255321	September 6, 2013	10-CURRENT after `gcc` and `libstdc++` are no longer built by default.
1000055	255449	September 6, 2013	10-CURRENT after addition of `MMAP_32BIT` mmap(2) flag (rev 255426).
1000100	259065	December 7, 2013	`releng/10.0` branched from `stable/10`.
1000500	256283	October 10, 2013	10-STABLE after branch from `head/`.
1000501	256916	October 22, 2013	10-STABLE after addition of first-boot rc(8) support.
1000502	258398	November 20, 2013	10-STABLE after removal of iconv symbols from `libc.so.7`.
1000510	259067	December 7, 2013	`releng/10.0` __FreeBSD_version update to prevent the value from going backwards.
1000700	259069	December 7, 2013	10-STABLE after `releng/10.0` branch.
1000701	259447	December 15, 2013	10.0-STABLE after Heimdal encoding fix.
1000702	260135	December 31, 2013	10-STABLE after MAP_STACK fixes.
1000703	262801	March 5, 2014	10-STABLE after upgrade of libc++ to 3.4 release.
1000704	262889	March 7, 2014	10-STABLE after MFC of the vt(4) driver (rev 262861).
1000705	263508	March 21, 2014	10-STABLE after upgrade of llvm/clang to 3.4 release.

Value	Revision	Date	Release
1000706	264214	April 6, 2014	10-STABLE after GCC support for `__block` definition.
1000707	264289	April 8, 2014	10-STABLE after FreeBSD-SA-14:06.openssl.
1000708	265122	April 30, 2014	10-STABLE after FreeBSD-SA-14:07.devfs, FreeBSD-SA-14:08.tcp, and FreeBSD-SA-14:09.openssl.
1000709	265946	May 13, 2014	10-STABLE after support for UDP-Lite protocol (RFC 3828).
1000710	267465	June 13, 2014	10-STABLE after changes to strcasecmp(3), moving strcasecmp_l(3) and strncasecmp_l(3) from <string.h> to <strings.h> for POSIX 2008 compliance.
1000711	268442	July 8, 2014	10-STABLE after FreeBSD-SA-14:17.kmem (rev 268432).
1000712	269400	August 1, 2014	10-STABLE after nfsd(8) 4.1 merge (rev 269398).
1000713	269484	August 3, 2014	10-STABLE after regex(3) library update to add ">" and "<" delimiters.
1000714	270174	August 3, 2014	10-STABLE after `SOCK_DGRAM` bug fix (rev 269490).
1000715	271341	September 9, 2014	10-STABLE after FreeBSD-SA-14:18 (rev 269686).
1000716	271686	September 16, 2014	10-STABLE after FreeBSD-SA-14:19 (rev 271667).

Value	Revision	Date	Release
1000717	271816	September 18, 2014	10-STABLE after i915 HW context support.
1001000	272463	October 2, 2014	10.1-RC1 after releng/10.1 branch.
1001500	272464	October 2, 2014	10-STABLE after releng/10.1 branch.
1001501	273432	October 21, 2014	10-STABLE after FreeBSD-SA-14:20, FreeBSD-SA-14:22, and FreeBSD-SA-14:23 (rev 273411).
1001502	274162	November 4, 2014	10-STABLE after FreeBSD-SA-14:23, FreeBSD-SA-14:24, and FreeBSD-SA-14:25.
1001503	275040	November 25, 2014	10-STABLE after merging new libraries/utilities (dpv(1) dpv(3), and figpar(3)) for data throughput visualization.
1001504	275742	December 13, 2014	10-STABLE after merging an important fix to the LLVM vectorizer, which could lead to buffer overruns in some cases.
1001505	276633	January 3, 2015	10-STABLE after merging some arm constants in 276312.
1001506	277087	January 12, 2015	10-STABLE after merging max table size update for yacc.
1001507	277790	January 27, 2015	10-STABLE after changes to the UDP tunneling callback to provide a context pointer and the source sockaddr.

Value	Revision	Date	Release
1001508	278974	February 18, 2015	10-STABLE after addition of the `CDAI_TYPE_EXT_INQ` request type.
1001509	279287	February 25, 2015	10-STABLE after FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp, and FreeBSD-SA-15:05.bind.
1001510	279329	February 26, 2015	10-STABLE after MFC of rev 278964.
1001511	280246	19 March, 2015	10-STABLE after sys/capability.h is renamed to sys/capsicum.h (rev 280224/).
1001512	280438	24 March, 2015	10-STABLE after addition of new mtio(4), sa(4) ioctls.
1001513	281955	24 April, 2015	10-STABLE after starting the process of removing the use of the deprecated "M_FLOWID" flag from the network code.
1001514	282275	April 30, 2015	10-STABLE after MFC of iconv(3) fixes.
1001515	282781	May 11, 2015	10-STABLE after adding back `M_FLOWID`.
1001516	283341	May 24, 2015	10-STABLE after MFC of many USB things.
1001517	283950	June 3, 2015	10-STABLE after MFC of sound related things.
1001518	284204	June 10, 2015	10-STABLE after MFC of zfs vfs fixes (rev 284203).
1001519	284720	June 23, 2015	10-STABLE after reverting bumping `MAXCPU` on amd64.

Value	Revision	Date	Release
1002000	285830	24 July, 2015	`releng/10.2` branched from 10-STABLE.
1002500	285831	24 July, 2015	10-STABLE after `releng/10.2` branched from 10-STABLE.
1002501	289005	8 October, 2015	10-STABLE after merge of ZFS changes that affected the internal interface of zfeature_info structure (rev 288572).
1002502	291243	24 November, 2015	10-STABLE after merge of dump device changes that affected the arguments of `g_dev_setdumpdev()`(rev 291215).
1002503	292224	14 December, 2015	10-STABLE after merge of changes to the internal interface between the nfsd.ko and nfscommon.ko modules, requiring them to be upgraded together (rev 292223).
1002504	292589	22 December, 2015	10-STABLE after merge of xz 5.2.2 merge (multithread support) (rev 292588).
1002505	292908	30 December, 2015	10-STABLE after merge of changes to pci(4) (rev 292907).
1002506	293476	9 January, 2016	10-STABLE after merge of utimensat(2) (rev 293473).
1002507	293610	9 January, 2016	10-STABLE after merge of changes to linux(4) (rev 293477 through 293609).

Value	Revision	Date	Release
1002508	293619	9 January, 2016	10-STABLE after merge of changes to figpar(3) types/macros (rev 290275).
1002509	295107	1 February, 2016	10-STABLE after merge of API change to dpv(3).
1003000	296373	4 March, 2016	`releng/10.3` branched from 10-STABLE.
1003500	296374	4 March, 2016	10-STABLE after `releng/10.3` branched from 10-STABLE.
1003501	298299	19 June, 2016	10-STABLE after adding kdbcontrol's -P option (rev 298297).
1003502	299966	19 June, 2016	10-STABLE after libcrypto.so was made position independent.
1003503	300235	19 June, 2016	10-STABLE after allowing MK_ overrides (rev 300233).
1003504	302066	21 June, 2016	10-STABLE after MFC of filemon changes from 11-CURRENT.
1003505	302228	27 June, 2016	10-STABLE after converting sed to use REG_STARTEND, fixing a Mesa issue.
1003506	304611	August 22, 2016	10-STABLE after adding C++11 thread_local support.
1003507	304864	August 26, 2016	10-STABLE after `LC_*_MASK` fix.
1003508	305734	September 12, 2016	10-STABLE after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).
1003509	307331	October 14, 2016	10-STABLE after ZFS merges.

Value	Revision	Date	Release
1003510	308047	October 28, 2016	10-STABLE after installing header files required development with libzfs_core.
1003511	310121	December 15, 2016	10-STABLE after exporting whole thread name in `kinfo_proc` (rev 309676).
1003512	315730	March 22, 2017	10-STABLE after libmd changes (rev 314143).
1003513	316499	April 4, 2017	10-STABLE after making CAM SIM lock optional (revs 315673, 315674).
1003514	318198	May 11, 2017	10-STABLE after merging the addition of the <dev/mmc/mmc_ioctl.h> header.
1003515	321222	July 19, 2017	10-STABLE after adding C14 sized deallocation functions to libc.
1003516	321717	July 30, 2017	10-STABLE after merging the `MAP_GUARD` mmap(2) flag addition.
1004000	323604	September 15, 2017	`releng/10.4` branched from 10-STABLE.
1004500	323605	September 15, 2017	10-STABLE after `releng/10.4` branched from 10-STABLE.
1004501	328379	January 24, 2018	10-STABLE after merging 325028, fixing `ptrace()` to always clear the correct thread event when resuming.
1004502	356396	January 6, 2020	10-STABLE after making USB statistics be per-device instead of per bus.

Value	Revision	Date	Release
1004503	356681	January 13, 2020	10-STABLE after adding own counter for cancelled USB transfers.

16.5. FreeBSD 9 Versions

Таблица 54. FreeBSD 9 `__FreeBSD_version` Values

Value	Revision	Date	Release
900000	196432	August 22, 2009	9.0-CURRENT.
900001	197019	September 8, 2009	9.0-CURRENT after importing x86emu, a software emulator for real mode x86 CPU from OpenBSD.
900002	197430	September 23, 2009	9.0-CURRENT after implementing the EVFILT_USER kevent filter functionality.
900003	200039	December 2, 2009	9.0-CURRENT after addition of sigpause(2) and PIE support in csu.
900004	200185	December 6, 2009	9.0-CURRENT after addition of libulog and its libutempter compatibility interface.
900005	200447	December 12, 2009	9.0-CURRENT after addition of sleepq_sleepcnt(9), which can be used to query the number of waiters on a specific waiting queue.
900006	201513	January 4, 2010	9.0-CURRENT after change of the scandir(3) and alphasort(3) prototypes to conform to SUSv4.

Value	Revision	Date	Release
900007	202219	January 13, 2010	9.0-CURRENT after the removal of utmp(5) and the addition of utmpx (see getutxent(3)) for improved logging of user logins and system events.
900008	202722	January 20, 2010	9.0-CURRENT after the import of BSDL bc/dc and the deprecation of GNU bc/dc.
900009	203052	January 26, 2010	9.0-CURRENT after the addition of SIOCGIFDESCR and SIOCSIFDESCR ioctls to network interfaces. These ioctl can be used to manipulate interface description, as inspired by OpenBSD.
900010	205471	March 22, 2010	9.0-CURRENT after the import of zlib 1.2.4.
900011	207410	April 24, 2010	9.0-CURRENT after adding soft-updates journalling.
900012	207842	May 10, 2010	9.0-CURRENT after adding liblzma, xz, xzdec, and lzmainfo.
900013	208486	May 24, 2010	9.0-CURRENT after bringing in USB fixes for linux(4).
900014	208973	June 10, 2010	9.0-CURRENT after adding Clang.
900015	210390	July 22, 2010	9.0-CURRENT after the import of BSD grep.
900016	210565	July 28, 2010	9.0-CURRENT after adding mti_zone to struct malloc_type_internal.

Value	Revision	Date	Release
900017	211701	August 23, 2010	9.0-CURRENT after changing back default grep to GNU grep and adding WITH_BSD_GREP knob.
900018	211735	August 24, 2010	9.0-CURRENT after the pthread_kill(3) -generated signal is identified as SI_LWP in si_code. Previously, si_code was SI_USER.
900019	211937	August 28, 2010	9.0-CURRENT after addition of the MAP_PREFAULT_READ flag to mmap(2).
900020	212381	September 9, 2010	9.0-CURRENT after adding drain functionality to sbufs, which also changed the layout of struct sbuf.
900021	212568	September 13, 2010	9.0-CURRENT after DTrace has grown support for userland tracing.
900022	213395	October 2, 2010	9.0-CURRENT after addition of the BSDL man utilities and retirement of GNU/GPL man utilities.
900023	213700	October 11, 2010	9.0-CURRENT after updating xz to git 20101010 snapshot.
900024	215127	November 11, 2010	9.0-CURRENT after libgcc.a was replaced by libcompiler_rt.a.
900025	215166	November 12, 2010	9.0-CURRENT after the introduction of the modularised congestion control.

Value	Revision	Date	Release
900026	216088	November 30, 2010	9.0-CURRENT after the introduction of Serial Management Protocol (SMP) passthrough and the XPT_SMP_IO and XPT_GDEV_ADVINFO CAM CCBs.
900027	216212	December 5, 2010	9.0-CURRENT after the addition of log2 to libm.
900028	216615	December 21, 2010	9.0-CURRENT after the addition of the Hhook (Helper Hook), Khelp (Kernel Helpers) and Object Specific Data (OSD) KPIs.
900029	216758	December 28, 2010	9.0-CURRENT after the modification of the TCP stack to allow Khelp modules to interact with it via helper hook points and store per-connection data in the TCP control block.
900030	217309	January 12, 2011	9.0-CURRENT after the update of libdialog to version 20100428.
900031	218414	February 7, 2011	9.0-CURRENT after the addition of pthread_getthreadid_np(3).
900032	218425	February 8, 2011	9.0-CURRENT after the removal of the uio_yield prototype and symbol.
900033	218822	February 18, 2011	9.0-CURRENT after the update of binutils to version 2.17.50.
900034	219406	March 8, 2011	9.0-CURRENT after the struct sysvec (sv_schedtail) changes.

Value	Revision	Date	Release
900035	220150	March 29, 2011	9.0-CURRENT after the update of base gcc and libstdc++ to the last GPLv2 licensed revision.
900036	220770	April 18, 2011	9.0-CURRENT after the removal of libobjc and Objective-C support from the base system.
900037	221862	May 13, 2011	9.0-CURRENT after importing the libprocstat(3) library and fuser(1) utility to the base system.
900038	222167	May 22, 2011	9.0-CURRENT after adding a lock flag argument to VFS_FHTOVP(9).
900039	223637	June 28, 2011	9.0-CURRENT after importing pf from OpenBSD 4.5.
900040	224217	July 19, 2011	Increase default MAXCPU for FreeBSD to 64 on amd64 and ia64 and to 128 for XLP (mips).
900041	224834	August 13, 2011	9.0-CURRENT after the implementation of Capsicum capabilities; fget(9) gains a rights argument.
900042	225350	August 28, 2011	Bump shared libraries' version numbers for libraries whose ABI has changed in preparation for 9.0.
900043	225350	September 2, 2011	Add automatic detection of USB mass storage devices which do not support the no synchronize cache SCSI command.

Value	Revision	Date	Release
900044	225469	September 10, 2011	Re-factor auto-quirk. 9.0-RELEASE.
900045	229285	January 2, 2012	9-STABLE after MFC of true/false from 1000002.
900500	229318	January 2, 2012	9.0-STABLE.
900501	229723	January 6, 2012	9.0-STABLE after merging of addition of the posix_fadvise(2) system call.
900502	230237	January 16, 2012	9.0-STABLE after merging gperf 3.0.3
900503	231768	February 15, 2012	9.0-STABLE after introduction of the new extensible sysctl(3) interface NET_RT_IFLISTL to query address lists.
900504	232728	March 3, 2012	9.0-STABLE after changes related to mounting of filesystem inside a jail.
900505	232945	March 13, 2012	9.0-STABLE after introduction of new tcp(4) socket options: TCP_KEEPINIT, TCP_KEEPIDLE, TCP_KEEPINTVL, and TCP_KEEPCNT.
900506	235786	May 22, 2012	9.0-STABLE after introduction of the `quick_exit` function and related changes required for C++11.
901000	239082	August 5, 2012	9.1-RELEASE.
901500	239081	August 6, 2012	9.1-STABLE after branching releng/9.1 (RELENG_9_1).

Value	Revision	Date	Release
901501	240659	November 11, 2012	9.1-STABLE after LIST_PREV(3) added to queue.h (rev 242893) and KBI change in USB serial devices.
901502	243656	November 28, 2012	9.1-STABLE after USB serial jitter buffer requires rebuild of USB serial device modules.
901503	247090	February 21, 2013	9.1-STABLE after USB moved to the driver structure requiring a rebuild of all USB modules. Also indicates the presence of nmtree.
901504	248338	March 15, 2013	9.1-STABLE after install gained -l, -M, -N and related flags and cat gained the -l option.
901505	251687	June 13, 2013	9.1-STABLE after fixes in ctfmerge bootstrapping (rev 249243).
902001	253912	August 3, 2013	`releng/9.2` branched from `stable/9`.
902501	253913	August 2, 2013	9.2-STABLE after creation of `releng/9.2` branch.
902502	254938	August 26, 2013	9.2-STABLE after inclusion of the `PIM_RESCAN` CAM path inquiry flag.
902503	254979	August 27, 2013	9.2-STABLE after inclusion of the `SI_UNMAPPED` cdev flag.
902504	256917	October 22, 2013	9.2-STABLE after inclusion of support for "first boot" rc(8) scripts.
902505	259448	December 12, 2013	9.2-STABLE after Heimdal encoding fix.

Value	Revision	Date	Release
902506	260136	December 31, 2013	9-STABLE after MAP_STACK fixes (rev 260082).
902507	262801	March 5, 2014	9-STABLE after upgrade of libc++ to 3.4 release.
902508	263171	March 14, 2014	9-STABLE after merge of the Radeon KMS driver (rev 263170).
902509	263509	March 21, 2014	9-STABLE after upgrade of llvm/clang to 3.4 release.
902510	263818	March 27, 2014	9-STABLE after merge of the vt(4) driver.
902511	264289	March 27, 2014	9-STABLE after FreeBSD-SA-14:06.openssl.
902512	265123	April 30, 2014	9-STABLE after FreeBSD-SA-14:08.tcp.
903000	267656	June 20, 2014	9-RC1 `releng/9.3` branch.
903500	267657	June 20, 2014	9.3-STABLE `releng/9.3` branch.
903501	268443	July 8, 2014	9-STABLE after FreeBSD-SA-14:17.kmem (rev 268433).
903502	270175	August 19, 2014	9-STABLE after `SOCK_DGRAM` bug fix (rev 269789).
903503	271341	September 9, 2014	9-STABLE after FreeBSD-SA-14:18 (rev 269687).
903504	271686	September 16, 2014	9-STABLE after FreeBSD-SA-14:19 (rev 271668).
903505	273432	October 21, 2014	9-STABLE after FreeBSD-SA-14:20, FreeBSD-SA-14:21, and FreeBSD-SA-14:22 (rev 273412).

Value	Revision	Date	Release
903506	274162	November 4, 2014	9-STABLE after FreeBSD-SA-14:23, FreeBSD-SA-14:24, and FreeBSD-SA-14:25.
903507	275742	December 13, 2014	9-STABLE after merging an important fix to the LLVM vectorizer, which could lead to buffer overruns in some cases.
903508	279287	February 25, 2015	9-STABLE after FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp, and FreeBSD-SA-15:05.bind.
903509	296219	February 29, 2016	9-STABLE after bumping the default value of `compat.linux.osrelease` to `2.6.18` to support the linux-c6-* ports out of the box.
903510	300236	May 19, 2016	9-STABLE after System Binary Interface (SBI) page was moved in latest version of Berkeley Boot Loader (BBL) due to code size increase in 300234.
903511	305735	September 12, 2016	9-STABLE after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).

16.6. FreeBSD 8 Versions

Таблица 55. FreeBSD 8 `__FreeBSD_version` Values

Value	Revision	Date	Release
800000	172531	October 11, 2007	8.0-CURRENT. Separating wide and single byte ctype.
800001	172688	October 16, 2007	8.0-CURRENT after libpcap 0.9.8 and tcpdump 3.9.8 import.
800002	172841	October 21, 2007	8.0-CURRENT after renaming kthread_create(9) and friends to kproc_create(9) etc.
800003	172932	October 24, 2007	8.0-CURRENT after ABI backwards compatibility to the FreeBSD 4/5/6 versions of the PCIOCGETCONF, PCIOCREAD and PCIOCWRITE IOCTLs was added, which required the ABI of the PCIOCGETCONF IOCTL to be broken again
800004	173573	November 12, 2007	8.0-CURRENT after agp(4) driver moved from src/sys/pci to src/sys/dev/agp
800005	174261	December 4, 2007	8.0-CURRENT after changes to the jumbo frame allocator (rev 174247).
800006	174399	December 7, 2007	8.0-CURRENT after the addition of callgraph capture functionality to hwpmc(4).
800007	174901	December 25, 2007	8.0-CURRENT after `kdb_enter()` gains a "why" argument.
800008	174951	December 28, 2007	8.0-CURRENT after LK_EXCLUPGRADE option removal.

Value	Revision	Date	Release
800009	175168	January 9, 2008	8.0-CURRENT after introduction of lockmgr_disown(9)
800010	175204	January 10, 2008	8.0-CURRENT after the vn_lock(9) prototype change.
800011	175295	January 13, 2008	8.0-CURRENT after the VOP_LOCK(9) and VOP_UNLOCK(9) prototype changes.
800012	175487	January 19, 2008	8.0-CURRENT after introduction of lockmgr_recursed(9), BUF_RECURSED(9) and BUF_ISLOCKED(9) and the removal of `BUF_REFCNT()`.
800013	175581	January 23, 2008	8.0-CURRENT after introduction of the "ASCII" encoding.
800014	175636	January 24, 2008	8.0-CURRENT after changing the prototype of lockmgr(9) and removal of `lockcount()` and `LOCKMGR_ASSERT()`.
800015	175688	January 26, 2008	8.0-CURRENT after extending the types of the fts(3) structures.
800016	175872	February 1, 2008	8.0-CURRENT after adding an argument to MEXTADD(9)
800017	176015	February 6, 2008	8.0-CURRENT after the introduction of LK_NODUP and LK_NOWITNESS options in the lockmgr(9) space.
800018	176112	February 8, 2008	8.0-CURRENT after the addition of m_collapse.

Value	Revision	Date	Release
800019	176124	February 9, 2008	8.0-CURRENT after the addition of current working directory, root directory, and jail directory support to the kern.proc.filedesc sysctl.
800020	176251	February 13, 2008	8.0-CURRENT after introduction of lockmgr_assert(9) and `BUF_ASSERT` functions.
800021	176321	February 15, 2008	8.0-CURRENT after introduction of lockmgr_args(9) and LK_INTERNAL flag removal.
800022	176556	(backed out)	8.0-CURRENT after changing the default system ar to BSD ar(1).
800023	176560	February 25, 2008	8.0-CURRENT after changing the prototypes of lockstatus(9) and VOP_ISLOCKED(9);, more specifically retiring the `struct thread` argument.
800024	176709	March 1, 2008	8.0-CURRENT after axing out the `lockwaiters` and `BUF_LOCKWAITERS` functions, changing the return value of `brelvp` from void to int and introducing new flags for lockinit(9).
800025	176958	March 8, 2008	8.0-CURRENT after adding F_DUP2FD command to fcntl(2).

Value	Revision	Date	Release
800026	177086	March 12, 2008	8.0-CURRENT after changing the priority parameter to cv_broadcastpri such that 0 means no priority.
800027	177551	March 24, 2008	8.0-CURRENT after changing the bpf monitoring ABI when zerocopy bpf buffers were added.
800028	177637	March 26, 2008	8.0-CURRENT after adding l_sysid to struct flock.
800029	177688	March 28, 2008	8.0-CURRENT after reintegration of the `BUF_LOCKWAITERS` function and the addition of lockmgr_waiters(9).
800030	177844	April 1, 2008	8.0-CURRENT after the introduction of the rw_try_rlock(9) and rw_try_wlock(9) functions.
800031	177958	April 6, 2008	8.0-CURRENT after the introduction of the `lockmgr_rw` and `lockmgr_args_rw` functions.
800032	178006	April 8, 2008	8.0-CURRENT after the implementation of the openat and related syscalls, introduction of the O_EXEC flag for the open(2), and providing the corresponding linux compatibility syscalls.

Value	Revision	Date	Release
800033	178017	April 8, 2008	8.0-CURRENT after added write(2) support for psm(4) in native operation level. Now arbitrary commands can be written to /dev/psm%d and status can be read back from it.
800034	178051	April 10, 2008	8.0-CURRENT after introduction of the `memrchr` function.
800035	178256	April 16, 2008	8.0-CURRENT after introduction of the `fdopendir` function.
800036	178362	April 20, 2008	8.0-CURRENT after switchover of 802.11 wireless to multi-bss support (aka vaps).
800037	178892	May 9, 2008	8.0-CURRENT after addition of multi routing table support (aka setfib(1), setfib(2)).
800038	179316	May 26, 2008	8.0-CURRENT after removal of netatm and ISDN4BSD. Also, the addition of the Compact C Type (CTF) tools.
800039	179784	June 14, 2008	8.0-CURRENT after removal of sgtty.
800040	180025	June 26, 2008	8.0-CURRENT with kernel NFS lockd client.
800041	180691	July 22, 2008	8.0-CURRENT after addition of arc4random_buf(3) and arc4random_uniform(3).
800042	181439	August 8, 2008	8.0-CURRENT after addition of cpuctl(4).

Value	Revision	Date	Release
800043	181694	August 13, 2008	8.0-CURRENT after changing bpf(4) to use a single device node, instead of device cloning.
800044	181803	August 17, 2008	8.0-CURRENT after the commit of the first step of the vimage project renaming global variables to be virtualized with a V_ prefix with macros to map them back to their global names.
800045	181905	August 20, 2008	8.0-CURRENT after the integration of the MPSAFE TTY layer, including changes to various drivers and utilities that interact with it.
800046	182869	September 8, 2008	8.0-CURRENT after the separation of the GDT per CPU on amd64 architecture.
800047	182905	September 10, 2008	8.0-CURRENT after removal of VSVTX, VSGID and VSUID.
800048	183091	September 16, 2008	8.0-CURRENT after converting the kernel NFS mount code to accept individual mount options in the nmount(2) iovec, not just one big struct nfs_args.
800049	183114	September 17, 2008	8.0-CURRENT after the removal of suser(9) and suser_cred(9).
800050	184099	October 20, 2008	8.0-CURRENT after buffer cache API change.

Value	Revision	Date	Release
800051	184205	October 23, 2008	8.0-CURRENT after the removal of the MALLOC(9) and FREE(9) macros.
800052	184419	October 28, 2008	8.0-CURRENT after the introduction of accmode_t and renaming of VOP_ACCESS 'a_mode' argument to 'a_accmode'.
800053	184555	November 2, 2008	8.0-CURRENT after the prototype change of vfs_busy(9) and the introduction of its MBF_NOWAIT and MBF_MNTLSTLOCK flags.
800054	185162	November 22, 2008	8.0-CURRENT after the addition of buf_ring, memory barriers and ifnet functions to facilitate multiple hardware transmit queues for cards that support them, and a lockless ring-buffer implementation to enable drivers to more efficiently manage queuing of packets.
800055	185363	November 27, 2008	8.0-CURRENT after the addition of Intel™ Core, Core2, and Atom support to hwpmc(4).
800056	185435	November 29, 2008	8.0-CURRENT after the introduction of multi-/no-IPv4/v6 jails.
800057	185522	December 1, 2008	8.0-CURRENT after the switch to the ath hal source code.

Value	Revision	Date	Release
800058	185968	December 12, 2008	8.0-CURRENT after the introduction of the VOP_VPTOCNP operation.
800059	186119	December 15, 2008	8.0-CURRENT incorporates the new arp-v2 rewrite.
800060	186344	December 19, 2008	8.0-CURRENT after the addition of makefs.
800061	187289	January 15, 2009	8.0-CURRENT after TCP Appropriate Byte Counting.
800062	187830	January 28, 2009	8.0-CURRENT after removal of minor(), minor2unit(), unit2minor(), etc.
800063	188745	February 18, 2009	8.0-CURRENT after GENERIC config change to use the USB2 stack, but also the addition of fdevname(3).
800064	188946	February 23, 2009	8.0-CURRENT after the USB2 stack is moved to and replaces dev/usb.
800065	189092	February 26, 2009	8.0-CURRENT after the renaming of all functions in libmp(3).
800066	189110	February 27, 2009	8.0-CURRENT after changing USB devfs handling and layout.
800067	189136	February 28, 2009	8.0-CURRENT after adding getdelim(), getline(), stpncpy(), strnlen(), wcsnlen(), wcscasecmp(), and wcsncasecmp().
800068	189276	March 2, 2009	8.0-CURRENT after renaming the ushub devclass to uhub.
800069	189585	March 9, 2009	8.0-CURRENT after libusb20.so.1 was renamed to libusb.so.1.

Value	Revision	Date	Release
800070	189592	March 9, 2009	8.0-CURRENT after merging IGMPv3 and Source-Specific Multicast (SSM) to the IPv4 stack.
800071	189825	March 14, 2009	8.0-CURRENT after gcc was patched to use C99 inline semantics in c99 and gnu99 mode.
800072	189853	March 15, 2009	8.0-CURRENT after the IFF_NEEDSGIANT flag has been removed; non-MPSAFE network device drivers are no longer supported.
800073	190265	March 18, 2009	8.0-CURRENT after the dynamic string token substitution has been implemented for rpath and needed paths.
800074	190373	March 24, 2009	8.0-CURRENT after tcpdump 4.0.0 and libpcap 1.0.0 import.
800075	190787	April 6, 2009	8.0-CURRENT after layout of structs vnet_net, vnet_inet and vnet_ipfw has been changed.
800076	190866	April 9, 2009	8.0-CURRENT after adding delay profiles in dummynet.
800077	190914	April 14, 2009	8.0-CURRENT after removing VOP_LEASE() and vop_vector.vop_lease.

Value	Revision	Date	Release
800078	191080	April 15, 2009	8.0-CURRENT after struct rt_weight fields have been added to struct rt_metrics and struct rt_metrics_lite, changing the layout of struct rt_metrics_lite. A bump to RTM_VERSION was made, but backed out.
800079	191117	April 15, 2009	8.0-CURRENT after struct llentry pointers are added to struct route and struct route_in6.
800080	191126	April 15, 2009	8.0-CURRENT after layout of struct inpcb has been changed.
800081	191267	April 19, 2009	8.0-CURRENT after the layout of struct malloc_type has been changed.
800082	191368	April 21, 2009	8.0-CURRENT after the layout of struct ifnet has changed, and with if_ref() and if_rele() ifnet refcounting.
800083	191389	April 22, 2009	8.0-CURRENT after the implementation of a low-level Bluetooth HCI API.
800084	191672	April 29, 2009	8.0-CURRENT after IPv6 SSM and MLDv2 changes.
800085	191688	April 30, 2009	8.0-CURRENT after enabling support for VIMAGE kernel builds with one active image.
800086	191910	May 8, 2009	8.0-CURRENT after adding support for input lines of arbitrarily length in patch(1).

Value	Revision	Date	Release
800087	191990	May 11, 2009	8.0-CURRENT after some VFS KPI changes. The thread argument has been removed from the FSD parts of the VFS. `VFS_*` functions do not need the context any more because it always refers to `curthread`. In some special cases, the old behavior is retained.
800088	192470	May 20, 2009	8.0-CURRENT after net80211 monitor mode changes.
800089	192649	May 23, 2009	8.0-CURRENT after adding UDP control block support.
800090	192669	May 23, 2009	8.0-CURRENT after virtualizing interface cloning.
800091	192895	May 27, 2009	8.0-CURRENT after adding hierarchical jails and removing global securelevel.
800092	193011	May 29, 2009	8.0-CURRENT after changing `sx_init_flags()` KPI. The `SX_ADAPTIVESPIN` is retired and a new `SX_NOADAPTIVE` flag is introduced to handle the reversed logic.
800093	193047	May 29, 2009	8.0-CURRENT after adding mnt_xflag to struct mount.
800094	193093	May 30, 2009	8.0-CURRENT after adding VOP_ACCESSX(9).

Value	Revision	Date	Release
800095	193096	May 30, 2009	8.0-CURRENT after changing the polling KPI. The polling handlers now return the number of packets processed. A new `IFCAP_POLLING_NOCOUNT` is also introduced to specify that the return value is not significant and the counting should be skipped.
800096	193219	June 1, 2009	8.0-CURRENT after updating to the new netisr implementation and after changing the way we store and access FIBs.
800097	193731	June 8, 2009	8.0-CURRENT after the introduction of vnet destructor hooks and infrastructure.
(not changed)	194012	June 11, 2009	8.0-CURRENT after the introduction of netgraph outbound to inbound path call detection and queuing, which also changed the layout of struct thread.
800098	194210	June 14, 2009	8.0-CURRENT after OpenSSL 0.9.8k import.
800099	194675	June 22, 2009	8.0-CURRENT after NGROUPS update and moving route virtualization into its own VImage module.
800100	194920	June 24, 2009	8.0-CURRENT after SYSVIPC ABI change.
800101	195175	June 29, 2009	8.0-CURRENT after the removal of the /dev/net/* per-interface character devices.

Value	Revision	Date	Release
800102	195634	July 12, 2009	8.0-CURRENT after padding was added to struct sackhint, struct tcpcb, and struct tcpstat.
800103	195654	July 13, 2009	8.0-CURRENT after replacing struct tcpopt with struct toeopt in the TOE driver interface to the TCP syncache.
800104	195699	July 14, 2009	8.0-CURRENT after the addition of the linker-set based per-vnet allocator.
800105	195767	July 19, 2009	8.0-CURRENT after version bump for all shared libraries that do not have symbol versioning turned on.
800106	195852	July 24, 2009	8.0-CURRENT after introduction of OBJT_SG VM object type.
800107	196037	August 2, 2009	8.0-CURRENT after making the newbus subsystem Giant free by adding the newbus sxlock and 8.0-RELEASE.
800108	199627	November 21, 2009	8.0-STABLE after implementing EVFILT_USER kevent filter.
800500	201749	January 7, 2010	8.0-STABLE after `__FreeBSD_version` bump to make `pkg_add -r` use packages-8-stable.

Value	Revision	Date	Release
800501	202922	January 24, 2010	8.0-STABLE after change of the scandir(3) and alphasort(3) prototypes to conform to SUSv4.
800502	203299	January 31, 2010	8.0-STABLE after addition of sigpause(2).
800503	204344	February 25, 2010	8.0-STABLE after addition of SIOCGIFDESCR and SIOCSIFDESCR ioctls to network interfaces. These ioctl can be used to manipulate interface description, as inspired by OpenBSD.
800504	204546	March 1, 2010	8.0-STABLE after MFC of importing x86emu, a software emulator for real mode x86 CPU from OpenBSD.
800505	208259	May 18, 2010	8.0-STABLE after MFC of adding liblzma, xz, xzdec, and lzmainfo.
801000	209150	June 14, 2010	8.1-RELEASE
801500	209146	June 14, 2010	8.1-STABLE after 8.1-RELEASE.
801501	214762	November 3, 2010	8.1-STABLE after KBI change in struct sysentvec, and implementation of PL_FLAG_SCE/SCX/EXEC/SI and pl_siginfo for ptrace(PT_LWPINFO) .
802000	216639	December 22, 2010	8.2-RELEASE
802500	216654	December 22, 2010	8.2-STABLE after 8.2-RELEASE.
802501	219107	February 28, 2011	8.2-STABLE after merging DTrace changes, including support for userland tracing.

Value	Revision	Date	Release
802502	219324	March 6, 2011	8.2-STABLE after merging log2 and log2f into libm.
802503	221275	May 1, 2011	8.2-STABLE after upgrade of the gcc to the last GPLv2 version from the FSF gcc-4_2-branch.
802504	222401	May 28, 2011	8.2-STABLE after introduction of the KPI and supporting infrastructure for modular congestion control.
802505	222406	May 28, 2011	8.2-STABLE after introduction of Hhook and Khelp KPIs.
802506	222408	May 28, 2011	8.2-STABLE after addition of OSD to struct tcpcb.
802507	222741	June 6, 2011	8.2-STABLE after ZFS v28 import.
802508	222846	June 8, 2011	8.2-STABLE after removal of the schedtail event handler and addition of the sv_schedtail method to struct sysvec.
802509	224017	July 14, 2011	8.2-STABLE after merging the SSSE3 support into binutils.
802510	224214	July 19, 2011	8.2-STABLE after addition of RFTSIGZMB flag for rfork(2).
802511	225458	September 9, 2011	8.2-STABLE after addition of automatic detection of USB mass storage devices which do not support the no synchronize cache SCSI command.

Value	Revision	Date	Release
802512	225470	September 10, 2011	8.2-STABLE after merging of re-factoring of auto-quirk.
802513	226763	October 25, 2011	8.2-STABLE after merging of the MAP_PREFAULT_READ flag to mmap(2).
802514	227573	November 16, 2011	8.2-STABLE after merging of addition of posix_fallocate(2) syscall.
802515	229725	January 6, 2012	8.2-STABLE after merging of addition of the posix_fadvise(2) system call.
802516	230239	January 16, 2012	8.2-STABLE after merging gperf 3.0.3
802517	231769	February 15, 2012	8.2-STABLE after introduction of the new extensible sysctl(3) interface NET_RT_IFLISTL to query address lists.
803000	232446	March 3, 2012	8.3-RELEASE.
803500	232439	March 3, 2012	8.3-STABLE after branching releng/8.3 (RELENG_8_3).
803501	247091	February 21, 2013	8.3-STABLE after MFC of two USB fixes (rev 246616 and 246759).
804000	248850	March 28, 2013	8.4-RELEASE.
804500	248819	March 28, 2013	8.4-STABLE after 8.4-RELEASE.
804501	259449	December 16, 2013	8.4-STABLE after MFC of upstream Heimdal encoding fix.
804502	265123	April 30, 2014	8.4-STABLE after FreeBSD-SA-14:08.tcp.
804503	268444	July 9, 2014	8.4-STABLE after FreeBSD-SA-14:17.kmem.

Value	Revision	Date	Release
804504	271341	September 9, 2014	8.4-STABLE after FreeBSD-SA-14:18 (rev 271305).
804505	271686	September 16, 2014	8.4-STABLE after FreeBSD-SA-14:19 (rev 271668).
804506	273432	October 21, 2014	8.4-STABLE after FreeBSD-SA-14:21 (rev 273413).
804507	274162	November 4, 2014	8.4-STABLE after FreeBSD-SA-14:23, FreeBSD-SA-14:24, and FreeBSD-SA-14:25.
804508	279287	February 25, 2015	8-STABLE after FreeBSD-EN-15:01.vt, FreeBSD-EN-15:02.openssl, FreeBSD-EN-15:03.freebsd-update, FreeBSD-SA-15:04.igmp, and FreeBSD-SA-15:05.bind.
804509	305736	September 12, 2016	8-STABLE after resolving a deadlock between `device_detach()` and usbd_do_request_flags(9).

16.7. FreeBSD 7 Versions

Таблица 56. FreeBSD 7 `__FreeBSD_version` Values

Value	Revision	Date	Release
700000	147925	July 11, 2005	7.0-CURRENT.
700001	148341	July 23, 2005	7.0-CURRENT after bump of all shared library versions that had not been changed since RELENG_5.
700002	149039	August 13, 2005	7.0-CURRENT after credential argument is added to dev_clone event handler.

Value	Revision	Date	Release
700003	149470	August 25, 2005	7.0-CURRENT after memmem(3) is added to libc.
700004	151888	October 30, 2005	7.0-CURRENT after solisten(9) kernel arguments are modified to accept a backlog parameter.
700005	152296	November 11, 2005	7.0-CURRENT after IFP2ENADDR() was changed to return a pointer to IF_LLADDR().
700006	152315	November 11, 2005	7.0-CURRENT after addition of `if_addr` member to `struct ifnet` and IFP2ENADDR() removal.
700007	153027	December 2, 2005	7.0-CURRENT after incorporating scripts from the local_startup directories into the base rcorder(8).
700008	153107	December 5, 2005	7.0-CURRENT after removal of MNT_NODEV mount option.
700009	153519	December 19, 2005	7.0-CURRENT after ELF-64 type changes and symbol versioning.
700010	153579	December 20, 2005	7.0-CURRENT after addition of hostb and vgapci drivers, addition of pci_find_extcap(), and changing the AGP drivers to no longer map the aperture.
700011	153936	December 31, 2005	7.0-CURRENT after tv_sec was made time_t on all platforms but Alpha.

Value	Revision	Date	Release
700012	154114	January 8, 2006	7.0-CURRENT after ldconfig_local_dirs change.
700013	154269	January 12, 2006	7.0-CURRENT after changes to /etc/rc.d/abi to support /compat/linux/etc/ld.so.cache being a symlink in a readonly filesystem.
700014	154863	January 26, 2006	7.0-CURRENT after pts import.
700015	157144	March 26, 2006	7.0-CURRENT after the introduction of version 2 of hwpmc(4)'s ABI.
700016	157962	April 22, 2006	7.0-CURRENT after addition of fcloseall(3) to libc.
700017	158513	May 13, 2006	7.0-CURRENT after removal of ip6fw.
700018	160386	July 15, 2006	7.0-CURRENT after import of snd_emu10kx.
700019	160821	July 29, 2006	7.0-CURRENT after import of OpenSSL 0.9.8b.
700020	161931	September 3, 2006	7.0-CURRENT after addition of bus_dma_get_tag function
700021	162023	September 4, 2006	7.0-CURRENT after libpcap 0.9.4 and tcpdump 3.9.4 import.
700022	162170	September 9, 2006	7.0-CURRENT after dlsym change to look for a requested symbol both in specified dso and its implicit dependencies.

Value	Revision	Date	Release
700023	162588	September 23, 2006	7.0-CURRENT after adding new sound IOCTLs for the OSSv4 mixer API.
700024	162919	September 28, 2006	7.0-CURRENT after import of OpenSSL 0.9.8d.
700025	164190	November 11, 2006	7.0-CURRENT after the addition of libelf.
700026	164614	November 26, 2006	7.0-CURRENT after major changes on sound sysctls.
700027	164770	November 30, 2006	7.0-CURRENT after the addition of Wi-Spy quirk.
700028	165242	December 15, 2006	7.0-CURRENT after the addition of sctp calls to libc
700029	166259	January 26, 2007	7.0-CURRENT after the GNU gzip(1) implementation was replaced with a BSD licensed version ported from NetBSD.
700030	166549	February 7, 2007	7.0-CURRENT after the removal of IPIP tunnel encapsulation (VIFF_TUNNEL) from the IPv4 multicast forwarding code.
700031	166907	February 23, 2007	7.0-CURRENT after the modification of bus_setup_intr() (newbus).
700032	167165	March 2, 2007	7.0-CURRENT after the inclusion of ipw(4) and iwi(4) firmware.
700033	167360	March 9, 2007	7.0-CURRENT after the inclusion of ncurses wide character support.

Value	Revision	Date	Release
700034	167684	March 19, 2007	7.0-CURRENT after changes to how insmntque(), getnewvnode(), and vfs_hash_insert() work.
700035	167906	March 26, 2007	7.0-CURRENT after addition of a notify mechanism for CPU frequency changes.
700036	168413	April 6, 2007	7.0-CURRENT after import of the ZFS filesystem.
700037	168504	April 8, 2007	7.0-CURRENT after addition of CAM 'SG' peripheral device, which implements a subset of Linux SCSI SG passthrough device API.
700038	169151	April 30, 2007	7.0-CURRENT after changing getenv(3), putenv(3), setenv(3) and unsetenv(3) to be POSIX conformant.
700039	169190	May 1, 2007	7.0-CURRENT after the changes in 700038 were backed out.
700040	169453	May 10, 2007	7.0-CURRENT after the addition of flopen(3) to libutil.
700041	169526	May 13, 2007	7.0-CURRENT after enabling symbol versioning, and changing the default thread library to libthr.
700042	169758	May 19, 2007	7.0-CURRENT after the import of gcc 4.2.0.
700043	169830	May 21, 2007	7.0-CURRENT after bump of all shared library versions that had not been changed since RELENG_6.

Value	Revision	Date	Release
700044	170395	June 7, 2007	7.0-CURRENT after changing the argument for vn_open()/VOP_OPEN() from file descriptor index to the struct file *.
700045	170510	June 10, 2007	7.0-CURRENT after changing pam_nologin(8) to provide an account management function instead of an authentication function to the PAM framework.
700046	170530	June 11, 2007	7.0-CURRENT after updated 802.11 wireless support.
700047	170579	June 11, 2007	7.0-CURRENT after adding TCP LRO interface capabilities.
700048	170613	June 12, 2007	7.0-CURRENT after RFC 3678 API support added to the IPv4 stack. Legacy RFC 1724 behavior of the IP_MULTICAST_IF ioctl has now been removed; 0.0.0.0/8 may no longer be used to specify an interface index. Use struct ipmreqn instead.
700049	171175	July 3, 2007	7.0-CURRENT after importing pf from OpenBSD 4.1
(not changed)	171167		7.0-CURRENT after adding IPv6 support for FAST_IPSEC, deleting KAME IPSEC, and renaming FAST_IPSEC to IPSEC.

Value	Revision	Date	Release
700050	171195	July 4, 2007	7.0-CURRENT after converting setenv/putenv/etc. calls from traditional BSD to POSIX.
700051	171211	July 4, 2007	7.0-CURRENT after adding new mmap/lseek/etc syscalls.
700052	171275	July 6, 2007	7.0-CURRENT after moving I4B headers to include/i4b.
700053	172394	September 30, 2007	7.0-CURRENT after the addition of support for PCI domains
700054	172988	October 25, 2007	7.0-STABLE after MFC of wide and single byte ctype separation.
700055	173104	October 28, 2007	7.0-RELEASE, and 7.0-CURRENT after ABI backwards compatibility to the FreeBSD 4/5/6 versions of the PCIOCGETCONF, PCIOCREAD and PCIOCWRITE IOCTLs was MFCed, which required the ABI of the PCIOCGETCONF IOCTL to be broken again
700100	174864	December 22, 2007	7.0-STABLE after 7.0-RELEASE
700101	176111	February 8, 2008	7.0-STABLE after the MFC of m_collapse().
700102	177735	March 30, 2008	7.0-STABLE after the MFC of kdb_enter_why().
700103	178061	April 10, 2008	7.0-STABLE after adding l_sysid to struct flock.
700104	178108	April 11, 2008	7.0-STABLE after the MFC of procstat(1).

Value	Revision	Date	Release
700105	178120	April 11, 2008	7.0-STABLE after the MFC of umtx features.
700106	178225	April 15, 2008	7.0-STABLE after the MFC of write(2) support to psm(4).
700107	178353	April 20, 2008	7.0-STABLE after the MFC of F_DUP2FD command to fcntl(2).
700108	178783	May 5, 2008	7.0-STABLE after some lockmgr(9) changes, which makes it necessary to include sys/lock.h to use lockmgr(9).
700109	179367	May 27, 2008	7.0-STABLE after MFC of the memrchr(3) function.
700110	181328	August 5, 2008	7.0-STABLE after MFC of kernel NFS lockd client.
700111	181940	August 20, 2008	7.0-STABLE after addition of physically contiguous jumbo frame support.
700112	182294	August 27, 2008	7.0-STABLE after MFC of kernel DTrace support.
701000	185315	November 25, 2008	7.1-RELEASE
701100	185302	November 25, 2008	7.1-STABLE after 7.1-RELEASE.
701101	187023	January 10, 2009	7.1-STABLE after strndup(3) merge.
701102	187370	January 17, 2009	7.1-STABLE after cpuctl(4) support added.
701103	188281	February 7, 2009	7.1-STABLE after the merge of multi-/no-IPv4/v6 jails.

Value	Revision	Date	Release
701104	188625	February 14, 2009	7.1-STABLE after the store of the suspension owner in the struct mount, and introduction of vfs_susp_clean method into the struct vfsops.
701105	189740	March 12, 2009	7.1-STABLE after the incompatible change to the kern.ipc.shmsegs sysctl to allow allocating larger SysV shared memory segments on 64bit architectures.
701106	189786	March 14, 2009	7.1-STABLE after the merge of a fix for POSIX semaphore wait operations.
702000	191099	April 15, 2009	7.2-RELEASE
702100	191091	April 15, 2009	7.2-STABLE after 7.2-RELEASE.
702101	192149	May 15, 2009	7.2-STABLE after ichsmb(4) was changed to use left-adjusted slave addressing to match other SMBus controller drivers.
702102	193020	May 28, 2009	7.2-STABLE after MFC of the fdopendir(3) function.
702103	193638	June 6, 2009	7.2-STABLE after MFC of PmcTools.
702104	195694	July 14, 2009	7.2-STABLE after MFC of the closefrom(2) system call.
702105	196006	July 31, 2009	7.2-STABLE after MFC of the SYSVIPC ABI change.

Value	Revision	Date	Release
702106	197198	September 14, 2009	7.2-STABLE after MFC of the x86 PAT enhancements and addition of d_mmap_single() and the scatter/gather list VM object type.
703000	203740	February 9, 2010	7.3-RELEASE
703100	203742	February 9, 2010	7.3-STABLE after 7.3-RELEASE.
704000	216647	December 22, 2010	7.4-RELEASE
704100	216658	December 22, 2010	7.4-STABLE after 7.4-RELEASE.
704101	221318	May 2, 2011	7.4-STABLE after the gcc MFC in rev 221317.

16.8. FreeBSD 6 Versions

Таблица 57. FreeBSD 6 `__FreeBSD_version` Values

Value	Revision	Date	Release
600000	133921	August 18, 2004	6.0-CURRENT
600001	134396	August 27, 2004	6.0-CURRENT after permanently enabling PFIL_HOOKS in the kernel.
600002	134514	August 30, 2004	6.0-CURRENT after initial addition of ifi_epoch to struct if_data. Backed out after a few days. Do not use this value.
600003	134933	September 8, 2004	6.0-CURRENT after the re-addition of the ifi_epoch member of struct if_data.
600004	135920	September 29, 2004	6.0-CURRENT after addition of the struct inpcb argument to the pfil API.

Value	Revision	Date	Release
600005	136172	October 5, 2004	6.0-CURRENT after addition of the "-d DESTDIR" argument to newsyslog.
600006	137192	November 4, 2004	6.0-CURRENT after addition of glibc style strftime(3) padding options.
600007	138760	December 12, 2004	6.0-CURRENT after addition of 802.11 framework updates.
600008	140809	January 25, 2005	6.0-CURRENT after changes to VOP_*VOBJECT() functions and introduction of MNTK_MPSAFE flag for Giantfree filesystems.
600009	141250	February 4, 2005	6.0-CURRENT after addition of the cpufreq framework and drivers.
600010	141394	February 6, 2005	6.0-CURRENT after importing OpenBSD's nc(1).
600011	141727	February 12, 2005	6.0-CURRENT after removing semblance of SVID2 `matherr()` support.
600012	141940	February 15, 2005	6.0-CURRENT after increase of default thread stacks' size.
600013	142089	February 19, 2005	6.0-CURRENT after fixes in <src/include/stdbool.h> and <src/sys/i386/include/_types.h> for using the GCC-compatibility of the Intel C/C++ compiler.

Value	Revision	Date	Release
600014	142184	February 21, 2005	6.0-CURRENT after EOVERFLOW checks in vswprintf(3) fixed.
600015	142501	February 25, 2005	6.0-CURRENT after changing the struct if_data member, ifi_epoch, from wall clock time to uptime.
600016	142582	February 26, 2005	6.0-CURRENT after LC_CTYPE disk format changed.
600017	142683	February 27, 2005	6.0-CURRENT after NLS catalogs disk format changed.
600018	142686	February 27, 2005	6.0-CURRENT after LC_COLLATE disk format changed.
600019	142752	February 28, 2005	Installation of acpica includes into /usr/include.
600020	143308	March 9, 2005	Addition of MSG_NOSIGNAL flag to send(2) API.
600021	143746	March 17, 2005	Addition of fields to cdevsw
600022	143901	March 21, 2005	Removed gtar from base system.
600023	144980	April 13, 2005	LOCAL_CREDS, LOCAL_CONNWAIT socket options added to unix(4).
600024	145565	April 19, 2005	hwpmc(4) and related tools added to 6.0-CURRENT.
600025	145565	April 26, 2005	struct icmphdr added to 6.0-CURRENT.
600026	145843	May 3, 2005	pf updated to 3.7.
600027	145966	May 6, 2005	Kernel libalias and ng_nat introduced.

Value	Revision	Date	Release
600028	146191	May 13, 2005	POSIX ttyname_r(3) made available through unistd.h and libc.
600029	146780	May 29, 2005	6.0-CURRENT after libpcap updated to v0.9.1 alpha 096.
600030	146988	June 5, 2005	6.0-CURRENT after importing NetBSD's if_bridge(4).
600031	147256	June 10, 2005	6.0-CURRENT after struct ifnet was broken out of the driver softcs.
600032	147898	July 11, 2005	6.0-CURRENT after the import of libpcap v0.9.1.
600033	148388	July 25, 2005	6.0-STABLE after bump of all shared library versions that had not been changed since RELENG_5.
600034	149040	August 13, 2005	6.0-STABLE after credential argument is added to dev_clone event handler. 6.0-RELEASE.
600100	151958	November 1, 2005	6.0-STABLE after 6.0-RELEASE
600101	153601	December 21, 2005	6.0-STABLE after incorporating scripts from the local_startup directories into the base rcorder(8).
600102	153912	December 30, 2005	6.0-STABLE after updating the ELF types and constants.
600103	154396	January 15, 2006	6.0-STABLE after MFC of pidfile(3) API.
600104	154453	January 17, 2006	6.0-STABLE after MFC of ldconfig_local_dirs change.

Value	Revision	Date	Release
600105	156019	February 26, 2006	6.0-STABLE after NLS catalog support of csh(1).
601000	158330	May 6, 2006	6.1-RELEASE
601100	158331	May 6, 2006	6.1-STABLE after 6.1-RELEASE.
601101	159861	June 22, 2006	6.1-STABLE after the import of csup.
601102	160253	July 11, 2006	6.1-STABLE after the iwi(4) update.
601103	160429	July 17, 2006	6.1-STABLE after the resolver update to BIND9, and exposure of reentrant version of netdb functions.
601104	161098	August 8, 2006	6.1-STABLE after DSO (dynamic shared objects) support has been enabled in OpenSSL.
601105	161900	September 2, 2006	6.1-STABLE after 802.11 fixups changed the api for the IEEE80211_IOC_STA_INFO ioctl.
602000	164312	November 15, 2006	6.2-RELEASE
602100	162329	September 15, 2006	6.2-STABLE after 6.2-RELEASE.
602101	165122	December 12, 2006	6.2-STABLE after the addition of Wi-Spy quirk.
602102	165596	December 28, 2006	6.2-STABLE after pci_find_extcap() addition.
602103	166039	January 16, 2007	6.2-STABLE after MFC of dlsym change to look for a requested symbol both in specified dso and its implicit dependencies.

Value	Revision	Date	Release
602104	166314	January 28, 2007	6.2-STABLE after MFC of ng_deflate(4) and ng_pred1(4) netgraph nodes and new compression and encryption modes for ng_ppp(4) node.
602105	166840	February 20, 2007	6.2-STABLE after MFC of BSD licensed version of gzip(1) ported from NetBSD.
602106	168133	March 31, 2007	6.2-STABLE after MFC of PCI MSI and MSI-X support.
602107	168438	April 6, 2007	6.2-STABLE after MFC of ncurses 5.6 and wide character support.
602108	168611	April 11, 2007	6.2-STABLE after MFC of CAM 'SG' peripheral device, which implements a subset of Linux SCSI SG passthrough device API.
602109	168805	April 17, 2007	6.2-STABLE after MFC of readline 5.2 patchset 002.
602110	169222	May 2, 2007	6.2-STABLE after MFC of pmap_invalidate_cache(), pmap_change_attr(), pmap_mapbios(), pmap_mapdev_attr(), and pmap_unmapbios() for amd64 and i386.
602111	170556	June 11, 2007	6.2-STABLE after MFC of BOP_BDFLUSH and caused breakage of the filesystem modules KBI.
602112	172284	September 21, 2007	6.2-STABLE after libutil(3) MFC's.

Value	Revision	Date	Release
602113	172986	October 25, 2007	6.2-STABLE after MFC of wide and single byte ctype separation. Newly compiled binary that references to ctype.h may require a new symbol, __mb_sb_limit, which is not available on older systems.
602114	173170	October 30, 2007	6.2-STABLE after ctype ABI forward compatibility restored.
602115	173794	November 21, 2007	6.2-STABLE after back out of wide and single byte ctype separation.
603000	173897	November 25, 2007	6.3-RELEASE
603100	173891	November 25, 2007	6.3-STABLE after 6.3-RELEASE.
(not changed)	174434	December 7, 2007	6.3-STABLE after fixing multibyte type support in bit macro.
603102	178459	April 24, 2008	6.3-STABLE after adding l_sysid to struct flock.
603103	179367	May 27, 2008	6.3-STABLE after MFC of the memrchr(3) function.
603104	179810	June 15, 2008	6.3-STABLE after MFC of support for `:u` variable modifier in make(1).
604000	183583	October 4, 2008	6.4-RELEASE
604100	183584	October 4, 2008	6.4-STABLE after 6.4-RELEASE.

16.9. FreeBSD 5 Versions

Таблица 58. FreeBSD 5 `__FreeBSD_version` Values

Value	Revision	Date	Release
500000	58009	March 13, 2000	5.0-CURRENT

Value	Revision	Date	Release
500001	59348	April 18, 2000	5.0-CURRENT after adding addition ELF header fields, and changing our ELF binary branding method.
500002	59906	May 2, 2000	5.0-CURRENT after kld metadata changes.
500003	60688	May 18, 2000	5.0-CURRENT after buf/bio changes.
500004	60936	May 26, 2000	5.0-CURRENT after binutils upgrade.
500005	61221	June 3, 2000	5.0-CURRENT after merging libxpg4 code into libc and after TASKQ interface introduction.
500006	61500	June 10, 2000	5.0-CURRENT after the addition of AGP interfaces.
500007	62235	June 29, 2000	5.0-CURRENT after Perl upgrade to 5.6.0
500008	62764	July 7, 2000	5.0-CURRENT after the update of KAME code to 2000/07 sources.
500009	63154	July 14, 2000	5.0-CURRENT after ether_ifattach() and ether_ifdetach() changes.
500010	63265	July 16, 2000	5.0-CURRENT after changing mtree defaults back to original variant, adding -L to follow symlinks.
500011	63459	July 18, 2000	5.0-CURRENT after kqueue API changed.
500012	65353	September 2, 2000	5.0-CURRENT after setproctitle(3) moved from libutil to libc.
500013	65671	September 10, 2000	5.0-CURRENT after the first SMPng commit.

Value	Revision	Date	Release
500014	70650	January 4, 2001	5.0-CURRENT after <sys/select.h> moved to <sys/selinfo.h>.
500015	70894	January 10, 2001	5.0-CURRENT after combining libgcc.a and libgcc_r.a, and associated GCC linkage changes.
500016	71583	January 24, 2001	5.0-CURRENT after change allowing libc and libc_r to be linked together, deprecating -pthread option.
500017	72650	February 18, 2001	5.0-CURRENT after switch from struct ucred to struct xucred to stabilize kernel-exported API for mountd et al.
500018	72975	February 24, 2001	5.0-CURRENT after addition of CPUTYPE make variable for controlling CPU-specific optimizations.
500019	77937	June 9, 2001	5.0-CURRENT after moving machine/ioctl_fd.h to sys/fdcio.h
500020	78304	June 15, 2001	5.0-CURRENT after locale names renaming.
500021	78632	June 22, 2001	5.0-CURRENT after Bzip2 import. Also signifies removal of S/Key.
500022	83435	July 12, 2001	5.0-CURRENT after SSE support.
500023	83435	September 14, 2001	5.0-CURRENT after KSE Milestone 2.
500024	84324	October 1, 2001	5.0-CURRENT after d_thread_t, and moving UUCP to ports.

Value	Revision	Date	Release
500025	84481	October 4, 2001	5.0-CURRENT after ABI change for descriptor and creds passing on 64 bit platforms.
500026	84710	October 9, 2001	5.0-CURRENT after moving to XFree86 4 by default for package builds, and after the new libc strnstr() function was added.
500027	84743	October 10, 2001	5.0-CURRENT after the new libc strcasestr() function was added.
500028	87879	December 14, 2001	5.0-CURRENT after the userland components of smbfs were imported.
(not changed)			5.0-CURRENT after the new C99 specific-width integer types were added.
500029	89938	January 29, 2002	5.0-CURRENT after a change was made in the return value of sendfile(2).
500030	90711	February 15, 2002	5.0-CURRENT after the introduction of the type `fflags_t`, which is the appropriate size for file flags.
500031	91203	February 24, 2002	5.0-CURRENT after the usb structure element rename.
500032	92453	March 16, 2002	5.0-CURRENT after the introduction of Perl 5.6.1.
500033	93722	April 3, 2002	5.0-CURRENT after the `sendmail_enable` rc.conf(5) variable was made to take the value `NONE`.

Value	Revision	Date	Release
500034	95831	April 30, 2002	5.0-CURRENT after mtx_init() grew a third argument.
500035	96498	May 13, 2002	5.0-CURRENT with Gcc 3.1.
500036	96781	May 17, 2002	5.0-CURRENT without Perl in /usr/src
500037	97516	May 29, 2002	5.0-CURRENT after the addition of dlfunc(3)
500038	100591	July 24, 2002	5.0-CURRENT after the types of some struct sockbuf members were changed and the structure was reordered.
500039	102757	September 1, 2002	5.0-CURRENT after GCC 3.2.1 import. Also after headers stopped using *BSD_FOO_T* and started using _FOO_T_DECLARED. This value can also be used as a conservative estimate of the start of bzip2(1) package support.
500040	103675	September 20, 2002	5.0-CURRENT after various changes to disk functions were made in the name of removing dependency on disklabel structure internals.
500041	104250	October 1, 2002	5.0-CURRENT after the addition of getopt_long(3) to libc.
500042	105178	October 15, 2002	5.0-CURRENT after Binutils 2.13 upgrade, which included new FreeBSD emulation, vec, and output format.

Value	Revision	Date	Release
500043	106289	November 1, 2002	5.0-CURRENT after adding weak pthread_XXX stubs to libc, obsoleting libXThrStub.so. 5.0-RELEASE.
500100	109405	January 17, 2003	5.0-CURRENT after branching for RELENG_5_0
500101	111120	February 19, 2003	<sys/dkstat.h> is empty. Do not include it.
500102	111482	February 25, 2003	5.0-CURRENT after the d_mmap_t interface change.
500103	111540	February 26, 2003	5.0-CURRENT after taskqueue_swi changed to run without Giant, and taskqueue_swi_giant added to run with Giant.
500104	111600	February 27, 2003	cdevsw_add() and cdevsw_remove() no longer exists. Appearance of MAJOR_AUTO allocation facility.
500105	111864	March 4, 2003	5.0-CURRENT after new cdevsw initialization method.
500106	112007	March 8, 2003	devstat_add_entry() has been replaced by devstat_new_entry()
500107	112288	March 15, 2003	Devstat interface change; see sys/sys/param.h 1.149
500108	112300	March 15, 2003	Token-Ring interface changes.
500109	112571	March 25, 2003	Addition of vm_paddr_t.

Value	Revision	Date	Release
500110	112741	March 28, 2003	5.0-CURRENT after realpath(3) has been made thread-safe
500111	113273	April 9, 2003	5.0-CURRENT after usbhid(3) has been synced with NetBSD
500112	113597	April 17, 2003	5.0-CURRENT after new NSS implementation and addition of POSIX.1 getpw*_r, getgr*_r functions
500113	114492	May 2, 2003	5.0-CURRENT after removal of the old rc system.
501000	115816	June 4, 2003	5.1-RELEASE.
501100	115710	June 2, 2003	5.1-CURRENT after branching for RELENG_5_1.
501101	117025	June 29, 2003	5.1-CURRENT after correcting the semantics of sigtimedwait(2) and sigwaitinfo(2).
501102	117191	July 3, 2003	5.1-CURRENT after adding the lockfunc and lockfuncarg fields to bus_dma_tag_create(9).
501103	118241	July 31, 2003	5.1-CURRENT after GCC 3.3.1-pre 20030711 snapshot integration.
501104	118511	August 5, 2003	5.1-CURRENT 3ware API changes to twe.
501105	119021	August 17, 2003	5.1-CURRENT dynamically-linked /bin and /sbin support and movement of libraries to /lib.
501106	119881	September 8, 2003	5.1-CURRENT after adding kernel support for Coda 6.x.

Value	Revision	Date	Release
501107	120180	September 17, 2003	5.1-CURRENT after 16550 UART constants moved from <dev/sio/sioreg.h> to <dev/ic/ns16550.h>. Also when libmap functionality was unconditionally supported by rtld.
501108	120386	September 23, 2003	5.1-CURRENT after PFIL_HOOKS API update
501109	120503	September 27, 2003	5.1-CURRENT after adding kiconv(3)
501110	120556	September 28, 2003	5.1-CURRENT after changing default operations for open and close in cdevsw
501111	121125	October 16, 2003	5.1-CURRENT after changed layout of cdevsw
501112	121129	October 16, 2003	5.1-CURRENT after adding kobj multiple inheritance
501113	121816	October 31, 2003	5.1-CURRENT after the if_xname change in struct ifnet
501114	122779	November 16, 2003	5.1-CURRENT after changing /bin and /sbin to be dynamically linked
502000	123198	December 7, 2003	5.2-RELEASE
502010	126150	February 23, 2004	5.2.1-RELEASE
502100	123196	December 7, 2003	5.2-CURRENT after branching for RELENG_5_2
502101	123677	December 19, 2003	5.2-CURRENT after *cxa_atexit*/cxa_finalize functions were added to libc.

Value	Revision	Date	Release
502102	125236	January 30, 2004	5.2-CURRENT after change of default thread library from libc_r to libpthread.
502103	126083	February 21, 2004	5.2-CURRENT after device driver API megapatch.
502104	126208	February 25, 2004	5.2-CURRENT after getopt_long_only() addition.
502105	126644	March 5, 2004	5.2-CURRENT after NULL is made into ((void *)0) for C, creating more warnings.
502106	126757	March 8, 2004	5.2-CURRENT after pf is linked to the build and install.
502107	126819	March 10, 2004	5.2-CURRENT after time_t is changed to a 64-bit value on sparc64.
502108	126891	March 12, 2004	5.2-CURRENT after Intel C/C++ compiler support in some headers and execve(2) changes to be more strictly conforming to POSIX.
502109	127312	March 22, 2004	5.2-CURRENT after the introduction of the bus_alloc_resource_any API
502110	127475	March 27, 2004	5.2-CURRENT after the addition of UTF-8 locales
502111	128144	April 11, 2004	5.2-CURRENT after the removal of the getvfsent(3) API
502112	128182	April 13, 2004	5.2-CURRENT after the addition of the .warning directive for make.

Value	Revision	Date	Release
502113	130057	June 4, 2004	5.2-CURRENT after ttyioctl() was made mandatory for serial drivers.
502114	130418	June 13, 2004	5.2-CURRENT after import of the ALTQ framework.
502115	130481	June 14, 2004	5.2-CURRENT after changing sema_timedwait(9) to return 0 on success and a non-zero error code on failure.
502116	130585	June 16, 2004	5.2-CURRENT after changing kernel dev_t to be pointer to struct cdev *.
502117	130640	June 17, 2004	5.2-CURRENT after changing kernel udev_t to dev_t.
502118	130656	June 17, 2004	5.2-CURRENT after adding support for CLOCK_VIRTUAL and CLOCK_PROF to clock_gettime(2) and clock_getres(2).
502119	130934	June 22, 2004	5.2-CURRENT after changing network interface cloning overhaul.
502120	131429	July 2, 2004	5.2-CURRENT after the update of the package tools to revision 20040629.
502121	131883	July 9, 2004	5.2-CURRENT after marking Bluetooth code as non-i386 specific.

Value	Revision	Date	Release
502122	131971	July 11, 2004	5.2-CURRENT after the introduction of the KDB debugger framework, the conversion of DDB into a backend and the introduction of the GDB backend.
502123	132025	July 12, 2004	5.2-CURRENT after change to make VFS_ROOT take a struct thread argument as does vflush. Struct kinfo_proc now has a user data pointer. The switch of the default X implementation to `xorg` was also made at this time.
502124	132597	July 24, 2004	5.2-CURRENT after the change to separate the way ports rc.d and legacy scripts are started.
502125	132726	July 28, 2004	5.2-CURRENT after the backout of the previous change.
502126	132914	July 31, 2004	5.2-CURRENT after the removal of kmem_alloc_pageable() and the import of gcc 3.4.2.
502127	132991	August 2, 2004	5.2-CURRENT after changing the UMA kernel API to allow ctors/inits to fail.
502128	133306	August 8, 2004	5.2-CURRENT after the change of the vfs_mount signature as well as global replacement of PRISON_ROOT with SUSER_ALLOWJAIL for the suser(9) API.

Value	Revision	Date	Release
503000	134189	August 23, 2004	5.3-BETA/RC before the pfil API change
503001	135580	September 22, 2004	5.3-RELEASE
503100	136595	October 16, 2004	5.3-STABLE after branching for RELENG_5_3
503101	138459	December 3, 2004	5.3-STABLE after addition of glibc style strftime(3) padding options.
503102	141788	February 13, 2005	5.3-STABLE after OpenBSD's nc(1) import MFC.
503103	142639	February 27, 2005	5.4-PRERELEASE after the MFC of the fixes in <src/include/stdbool.h> and <src/sys/i386/include/_types.h> for using the GCC-compatibility of the Intel C/C++ compiler.
503104	142835	February 28, 2005	5.4-PRERELEASE after the MFC of the change of ifi_epoch from wall clock time to uptime.
503105	143029	March 2, 2005	5.4-PRERELEASE after the MFC of the fix of EOVERFLOW check in vswprintf(3).
504000	144575	April 3, 2005	5.4-RELEASE.
504100	144581	April 3, 2005	5.4-STABLE after branching for RELENG_5_4
504101	146105	May 11, 2005	5.4-STABLE after increasing the default thread stacksizes
504102	504101	June 24, 2005	5.4-STABLE after the addition of sha256
504103	150892	October 3, 2005	5.4-STABLE after the MFC of if_bridge

Value	Revision	Date	Release
504104	152370	November 13, 2005	5.4-STABLE after the MFC of bsdiff and portsnap
504105	154464	January 17, 2006	5.4-STABLE after MFC of ldconfig_local_dirs change.
505000	158481	May 12, 2006	5.5-RELEASE.
505100	158482	May 12, 2006	5.5-STABLE after branching for RELENG_5_5

16.10. FreeBSD 4 Versions

Таблица 59. FreeBSD 4 `__FreeBSD_version` Values

Value	Revision	Date	Release
400000	43041	January 22, 1999	4.0-CURRENT after 3.4 branch
400001	44177	February 20, 1999	4.0-CURRENT after change in dynamic linker handling
400002	44699	March 13, 1999	4.0-CURRENT after C++ constructor/destructor order change
400003	45059	March 27, 1999	4.0-CURRENT after functioning dladdr(3)
400004	45321	April 5, 1999	4.0-CURRENT after __deregister_frame_info dynamic linker bug fix (also 4.0-CURRENT after EGCS 1.1.2 integration)
400005	46113	April 27, 1999	4.0-CURRENT after suser(9) API change (also 4.0-CURRENT after newbus)
400006	47640	May 31, 1999	4.0-CURRENT after cdevsw registration change
400007	47992	June 17, 1999	4.0-CURRENT after the addition of so_cred for socket level credentials

Value	Revision	Date	Release
400008	48048	June 20, 1999	4.0-CURRENT after the addition of a poll syscall wrapper to libc_r
400009	48936	July 20, 1999	4.0-CURRENT after the change of the kernel's `dev_t` type to `struct specinfo` pointer
400010	51649	September 25, 1999	4.0-CURRENT after fixing a hole in jail(2)
400011	51791	September 29, 1999	4.0-CURRENT after the `sigset_t` datatype change
400012	53164	November 15, 1999	4.0-CURRENT after the cutover to the GCC 2.95.2 compiler
400013	54123	December 4, 1999	4.0-CURRENT after adding pluggable linux-mode ioctl handlers
400014	56216	January 18, 2000	4.0-CURRENT after importing OpenSSL
400015	56700	January 27, 2000	4.0-CURRENT after the C++ ABI change in GCC 2.95.2 from -fvtable -thunks to -fno-vtable -thunks by default
400016	57529	February 27, 2000	4.0-CURRENT after importing OpenSSH
400017	58005	March 13, 2000	4.0-RELEASE
400018	58170	March 17, 2000	4.0-STABLE after 4.0-RELEASE
400019	60047	May 5, 2000	4.0-STABLE after the introduction of delayed checksums.
400020	61262	June 4, 2000	4.0-STABLE after merging libxpg4 code into libc.

Value	Revision	Date	Release
400021	62820	July 8, 2000	4.0-STABLE after upgrading Binutils to 2.10.0, ELF branding changes, and tcsh in the base system.
410000	63095	July 14, 2000	4.1-RELEASE
410001	64012	July 29, 2000	4.1-STABLE after 4.1-RELEASE
410002	65962	September 16, 2000	4.1-STABLE after setproctitle(3) moved from libutil to libc.
411000	66336	September 25, 2000	4.1.1-RELEASE
411001			4.1.1-STABLE after 4.1.1-RELEASE
420000	68066	October 31, 2000	4.2-RELEASE
420001	70895	January 10, 2001	4.2-STABLE after combining libgcc.a and libgcc_r.a, and associated GCC linkage changes.
430000	73800	March 6, 2001	4.3-RELEASE
430001	76779	May 18, 2001	4.3-STABLE after wint_t introduction.
430002	80157	July 22, 2001	4.3-STABLE after PCI powerstate API merge.
440000	80923	August 1, 2001	4.4-RELEASE
440001	85341	October 23, 2001	4.4-STABLE after d_thread_t introduction.
440002	86038	November 4, 2001	4.4-STABLE after mount structure changes (affects filesystem klds).
440003	88130	December 18, 2001	4.4-STABLE after the userland components of smbfs were imported.
450000	88271	December 20, 2001	4.5-RELEASE
450001	91203	February 24, 2002	4.5-STABLE after the usb structure element rename.

Value	Revision	Date	Release
450002	92151	March 12, 2002	4.5-STABLE after locale changes.
450003			(Never created)
450004	94840	April 16, 2002	4.5-STABLE after the `sendmail_enable` rc.conf(5) variable was made to take the value `NONE`.
450005	95555	April 27, 2002	4.5-STABLE after moving to XFree86 4 by default for package builds.
450006	95846	May 1, 2002	4.5-STABLE after accept filtering was fixed so that is no longer susceptible to an easy DoS.
460000	97923	June 21, 2002	4.6-RELEASE
460001	98730	June 21, 2002	4.6-STABLE sendfile(2) fixed to comply with documentation, not to count any headers sent against the amount of data to be sent from the file.
460002	100366	July 19, 2002	4.6.2-RELEASE
460100	98857	June 26, 2002	4.6-STABLE
460101	98880	June 26, 2002	4.6-STABLE after MFC of `sed -i`.
460102	102759	September 1, 2002	4.6-STABLE after MFC of many new pkg_install features from the HEAD.
470000	104655	October 8, 2002	4.7-RELEASE
470100	104717	October 9, 2002	4.7-STABLE

Value	Revision	Date	Release
470101	106732	November 10, 2002	Start generated *std{in,out,err}p references rather than* sF. This changes std{in,out,err} from a compile time expression to a runtime one.
470102	109753	January 23, 2003	4.7-STABLE after MFC of mbuf changes to replace m_aux mbufs by m_tag's
470103	110887	February 14, 2003	4.7-STABLE gets OpenSSL 0.9.7
480000	112852	March 30, 2003	4.8-RELEASE
480100	113107	April 5, 2003	4.8-STABLE
480101	115232	May 22, 2003	4.8-STABLE after realpath(3) has been made thread-safe
480102	118737	August 10, 2003	4.8-STABLE 3ware API changes to twe.
490000	121592	October 27, 2003	4.9-RELEASE
490100	121593	October 27, 2003	4.9-STABLE
490101	124264	January 8, 2004	4.9-STABLE after e_sid was added to struct kinfo_eproc.
490102	125417	February 4, 2004	4.9-STABLE after MFC of libmap functionality for rtld.
491000	129700	May 25, 2004	4.10-RELEASE
491100	129918	June 1, 2004	4.10-STABLE
491101	133506	August 11, 2004	4.10-STABLE after MFC of revision 20040629 of the package tools
491102	137786	November 16, 2004	4.10-STABLE after VM fix dealing with unwiring of fictitious pages
492000	138960	December 17, 2004	4.11-RELEASE
492100	138959	December 17, 2004	4.11-STABLE

Value	Revision	Date	Release
492101	157843	April 18, 2006	4.11-STABLE after adding libdata/ldconfig directories to mtree files.

16.11. FreeBSD 3 Versions

Таблица 60. FreeBSD 3 `__FreeBSD_version` Values

Value	Revision	Date	Release
300000	22917	February 19, 1996	3.0-CURRENT before mount(2) change
300001	36283	September 24, 1997	3.0-CURRENT after mount(2) change
300002	36592	June 2, 1998	3.0-CURRENT after semctl(2) change
300003	36735	June 7, 1998	3.0-CURRENT after ioctl arg changes
300004	38768	September 3, 1998	3.0-CURRENT after ELF conversion
300005	40438	October 16, 1998	3.0-RELEASE
300006	40445	October 16, 1998	3.0-CURRENT after 3.0-RELEASE
300007	43042	January 22, 1999	3.0-STABLE after 3/4 branch
310000	43807	February 9, 1999	3.1-RELEASE
310001	45060	March 27, 1999	3.1-STABLE after 3.1-RELEASE
310002	45689	April 14, 1999	3.1-STABLE after C++ constructor/destructor order change
320000			3.2-RELEASE
320001	46742	May 8, 1999	3.2-STABLE
320002	50563	August 29, 1999	3.2-STABLE after binary-incompatible IPFW and socket changes
330000	50813	September 2, 1999	3.3-RELEASE
330001	51328	September 16, 1999	3.3-STABLE

Value	Revision	Date	Release
330002	53671	November 24, 1999	3.3-STABLE after adding mkstemp(3) to libc
340000	54166	December 5, 1999	3.4-RELEASE
340001	54730	December 17, 1999	3.4-STABLE
350000	61876	June 20, 2000	3.5-RELEASE
350001	63043	July 12, 2000	3.5-STABLE

16.12. FreeBSD 2.2 Versions

Таблица 61. FreeBSD 2.2 `__FreeBSD_version` Values

Value	Revision	Date	Release
220000	22918	February 19, 1997	2.2-RELEASE
(not changed)			2.2.1-RELEASE
(not changed)			2.2-STABLE after 2.2.1-RELEASE
221001	24941	April 15, 1997	2.2-STABLE after texinfo-3.9
221002	25325	April 30, 1997	2.2-STABLE after top
222000	25851	May 16, 1997	2.2.2-RELEASE
222001	25921	May 19, 1997	2.2-STABLE after 2.2.2-RELEASE
225000	30053	October 2, 1997	2.2.5-RELEASE
225001	31300	November 20, 1997	2.2-STABLE after 2.2.5-RELEASE
225002	32019	December 27, 1997	2.2-STABLE after ldconfig -R merge
226000	34445	March 24, 1998	2.2.6-RELEASE
227000	37803	July 21, 1998	2.2.7-RELEASE
227001	37809	July 21, 1998	2.2-STABLE after 2.2.7-RELEASE
227002	39489	September 19, 1998	2.2-STABLE after semctl(2) change
228000	41403	November 29, 1998	2.2.8-RELEASE
228001	41418	November 29, 1998	2.2-STABLE after 2.2.8-RELEASE

Note that 2.2-STABLE sometimes identifies itself as "2.2.5-STABLE" after the 2.2.5-RELEASE. The pattern used to be year followed by the month, but we decided to change it to a more straightforward major/minor system starting from 2.2. This is because the parallel development on several branches made it infeasible to classify the releases merely by their real release dates. Do not worry about old -CURRENTs; they are listed here just for reference.

16.13. FreeBSD 2 Before 2.2-RELEASE Versions

Таблица 62. FreeBSD 2 Before 2.2-RELEASE `__FreeBSD_version` Values

Value	Revision	Date	Release
119411			2.0-RELEASE
199501	7153	March 19, 1995	2.1-CURRENT
199503	7310	March 24, 1995	2.1-CURRENT
199504	7704	April 9, 1995	2.0.5-RELEASE
199508	10297	August 26, 1995	2.2-CURRENT before 2.1
199511	12189	November 10, 1995	2.1.0-RELEASE
199512	12196	November 10, 1995	2.2-CURRENT before 2.1.5
199607	17067	July 10, 1996	2.1.5-RELEASE
199608	17127	July 12, 1996	2.2-CURRENT before 2.1.6
199612	19358	November 15, 1996	2.1.6-RELEASE
199612			2.1.7-RELEASE